Crystal reports 2008 case statement

I have a switch statement in a Crystal Report that looks like this: stringvar tag := {table1.field1}; Includes a string value selected item labeled 'first value': {table1.field3} 'second value': {table4.field9} Default: 'Unknown'; But when I try to save the formula, I get this error: ---------- crystal reports ---------- the remaining text does not appear to be part of the formula. ---------- good ---------- and it highlights the starting formula with quotes only before the second word in my example. What stupid syntax error am I doing? One complaint that is sometimes heard in the competitive community of database reporting authors is that, Crystal's reports are too complicated it's made for programmers. While the complaint may or may not ring correctly, there is no doubt that elements of common programming languages can be found in Crystal Reports formula languages. The first of these programming-oriented features is Logic If-Then-Else in formulas. Combining If-Then-Else is the cornerstone of much computer programming code, so once you learn the concepts of If-Then-Else, you will be really complicated on your way to doing reporting customization. If-Then-Else formulas perform a test on a database field, another formula, or a combination of them. Your test can be as simple or as complicated as you need it to perhaps just check out to see if a sales figure exceeds the bonus threshold of $1,000. Or, you may want to check the number of days a product took to the ship, in conjunction with the carrier that carries the product and the customer's sales level, to determine whether a shipment meets your company's shipping targets. If the test passes (it returns correctly), the formula returns a certain result. If the test fails (incorrectly returns), another result will be returned. If-Then-Else formulas are created with the following syntax: If <test>Then The test part of an <result if= true=> <result if= false=>If-Then-Else formula must use comparison operators found in the Operator Tree box (or a Boolean formula, discussed later in the chapter). You'll find a comparison section of the box that, when opened, shows operators who test for equal, less than, more than, and other combinations of conditions. These can be used in conjunction with operators And, Or, and Not Boolean to combine multiple conditional tests together. Here is a simple If-Then-Else formula that returns a string based on the order value: if {Orders.Order Amount} > 5000 then the bonus order is another regular order of the database field the order value is tested to see if it is worth more than 5000. If the test is correct, the string formula returns the bonus command. Otherwise the formula returns the order string regularly. Boolean operators can also be used to combine multiple comparisons together. You can use and, or, not Boolean operators. The previous formula has been slightly enhanced in the following formula, using a</result > </result> </test>Operator to combine two comparisons: If {Orders.Order value} > 5000 and month ({Orders.Order Date}) = 12 then the holiday bonus order is no longer a regular order here, the order value must be more than 5,000 and the order must be placed in December for the formula for returning the holiday bonus order. Orders of more than 5,000 in other months will still be regular orders. If you change to or in the previous formula, then all orders in December will reward orders, regardless of the amount. Orders over 5,000 will also be considered bonus orders the rest of the year. When creating If-Then-Else formulas, you should pay special attention to the types of data you use in the formula. In the If Formula test, make sure you use the same data types in each individual comparison operation. For example, if you want to test whether the customer.country is USA, the test will be if {Customer.Country} = THE USA of Jupiter.Country is a field string, you need to compare it to a literal string, enclosed in quotation marks or apostrophes (only quotation marks in the basic syntax). If the field you are testing is numerical, you should use a fixed number, as shown in the Orders.Order Amount sample as shown earlier. If you mismatch the data type, like these: if you get {Orders.Order Amount} > 5000 errors. If you use multiple isolated comparisons by Boolean operators, each comparison can have a different data type. For example, if you want to combine the two tests listed earlier, your formula starts as follows: If {Client.Country} = USA and {Orders.Order Amount} > 5000 in this case, different data types in if part of the formula is good, as long as each side of each comparison is the same data type. For example, you may have a formula in your report, @Ship Days, which calculates the number of days it took to ship an order. But from @Ship day is a numeric formula, if the order is placed and shipped on the same day, it will display zeros on your report. So, you will find the following If-Then-Else formula to show the same day words in the report if @Ship days are zero, or to show only the contents of formula @Ship days if not zero: if {@Ship days} = 0 then the same day other {@Ship days} but if you use the check button in the formula editor to check the syntax of this formula, you will get an error: the problem is that the crystal report does not know what type of data to assign to the formula. If the test returns correctly, formula one string will return for words the same day. However, if the wrong test returns, the formula @Ship to the number that is a number. Crystal Report should assign a data type to a formula when it is first created it can't wait to report running. So, even if part of a formula can contain different data type tests, then and other sections should include the same data types. Remember the function that converts other data types to strings? The following will be solved problem: If {@Ship Days} = 0 Then Same Day Else ToText({@Ship Days},0) This result is better, because it doesn't show zero as the number of ship days. But we want to take it one step further to make the report more readable. Look at the advanced version of this formula: If {@Ship Days} = 0 then Shipped Same Day Else Shipped in + ToText({@Ship Days},0) + Days This looks better on the report, particularly if the report isn't a straight columnar report. You may want to put this in a group header to order, before individual line items for order are shown in the Details section. But there is only one more problem. If it took only one day for the ship's order to be taken, what would this formula go back to? Carrying in 1 day while this probably doesn't mean your dismissal from the report's development team, why take an easier step further to report looking even better? Try the following: If {@Ship Days} = 0 Then Shipped Same Day Else If {@Ship Days} = 1 Then Shipped in 1 day Else Shipped in + ToText({@Ship Days},0) + These days is a example of a compound or nested If-Then-Else statement. Note that you are not limited to one if, then, or another paragraph in a formula. You can make another if the first result statement then, or the other first result, and on and on. There are no specific limitations to how many levels you can nest this, but obviously the formula gets hard to follow after only one or two levels. Also, don't forget that else clause is not required, so once you nest this, you don't have to always have another matching clause for any if. You will notice in all previous instances that only one action occurred as a result of the Then and Else sections of the statement. While this is not a problem for many types of formulas, sometimes you may want a few things to happen, especially when you need to set multiple variable contents as the results of a single Then or Else clause (variables are discussed later in this chapter). In these situations, you can simply repeat the If-Then-Else test several times, with a different result for each Then and Else clause. Just be sure to separate any If-Then-Else statement from next with semicolon. For example, to set multiple variable contents in a formula, you may use the following: NumberVar GroupBonus; NumberVar ReportBonus; NumberVar ReportFollowUpcount; StringVar GoodCustomer; If {Orders.Order Amount} > 5000 Then GroupBonus := GroupBonus + 1 Else GroupFollowUpCount := GroupFollowUpCount + 1 ; If {Orders.Order Amount} > 5000 Then ReportBonus := ReportBonus + 1 Else ReportFollowUpCount := ReportFollowUpCount + 1 ; If {Orders.Order Amount} > 5000 Then GoodCustomer := {Customer.Customer Name} All of these statements will be evaluated and the resulting variables will be set. The formula will display the result of the latest action in the If statement on the report. In this example, if the order amount exceeds $5,000, all bonuses By one boost will be variable GoodCustomer will be assigned customer name. But since goodCustomer variable assignment is the last action it runs, the customer name is what the formula actually displays on the report. If the order value is less than $5,000, then the two FollowUpCount variables will be increased by one. But since the latest statement still tries to adjust the GoodCustomer variable and fails (and there is no other clause), the GoodCustomer variable will return the assigned value and formula one blank string. If you look at the previous example, you'll see quite a little duplicate typing (if test repeats three times). This duplication can be removed by creating only one If-Then-Else statement, but supplies several actions to the then and other clauses, separating statements of action with semicolon and surrounding them all with pranat. Here is the same formula created using this shortened approach: NumberVar GroupBonus; NumberVar GroupFollowUpCount; NumberVar ReportBonus; NumberVar ReportFollowUpcount; StringVar GoodCustomer; If {Orders.Order Amount} > 5000 Then (GroupBonus := GroupBonus + 1 ; ReportBonus := ReportBonus +1 ; GoodCustomer := {Customer.Customer Name}) Else (GroupFollowUpCount := GroupFollowUpCount + 1 ; ReportFollowUpCount := ReportFollowUpCount + 1; GoodCustomer := Caution in the previous example, you should make sure to include a statement to assign the GoodCustomer variable an empty string in the Else clause. Otherwise, you will get a string needed here error. This is because then and Else clauses still need to return the same type of data. Without assigning GoodCustomer in the Else clause, then paragraph will be returning the string value (a string variable being assigned), and the other paragraph will return the numeric value (a numeric variable is being assigned). If you look at the Formula Editor function tree box, you will notice a print mode category. Opening this category shows the types of built-in functions that you can use in If-Then-Else (and other) formulas to increase the flexibility of your report. For example, all special fields discussed in Chapter 2, such as page number, total page number, print and time date, record number, group number, and others are available. There are other specific functions that you can use to test for the null database value now, next, or last record; With these built-in specific functions, you can create formulas that will make your reports more intuitive and easier to read. Advanced Crystal Reports If-Then-Other Options If-Then-Another Logic Described So Far Is Not Enough To Take You Towards (or Perhaps Away From) Professional Programming, Crystal Reports includes bevy of other if-then-other possibilities to help you reconsider! First Statements are described in a fundamentally different syntax than those crystal syntax versions previously. The basic syntax follows a more typical If-Then-Else-EndIf approach familiar to basic language programmers. In particular, this makes performing multiple acts as the result of a single If-Then-Else statement more directly by introducing the end clause if: if it then no longer ends if also, you can use the ElseIf basic syntax clause (don't forget to make it a word without space) allowing the toto of several conditions if in a statement: if then ElseIf then finish if you miss another paste of Logic <test> <statement> <statement> >lt;more statements=> > <statement> >lt;more statements=> <first test=> <statements> >lt;second test=> > lt;second test= statements=> <statements if= first= test= fails=>If-Then-Else exists in both crystalline and basic syntax. If you've used Microsoft Office products like Microsoft Access, you may be familiar with the IIF Immediate If function. This shortened version of If-Then-Else's Long Logic is actually a single function, similar to ToText or UpperCase (appears in the function tree under programming shortcuts) that accepts three argyms. The function syntax is as follows: IIF(,) This function can simplify If-Then-Else logic for simple, small and simple formulas, or when you want to make a Mini<boolean expression statement=>lt;result if=true=>If-Then-Else as part of a larger formula <result if= false=>. For example, consider the following string formula using the traditional If-Then-Else logic: If {Customer.Country} = USA Then {Customer.Customer Name} + requires domestic shipping costs {Customer.Customer Name} + requires international shipping costs using the Immediate If function, this can be simplified to the following: {Customer.Customer Name} + requires + IIF (Customer{Customer.Country} = USA, domestic, international) + shipping charges Here are some examples: IsNull function If IsNull({Customer.Region}) Then Else, + {Customer.Region} The IsNull function is critical if you may encounter null values in database fields that you include in your formulas. With the design, a crystal reporting formula returns an empty value if each part of the field contains a null value. If an added numeric formula is performed on the field containing the empty value, the formula will not treat the null as zero, and return a numeric result with the rest of the added formula one null. If you are doing concatenation string and one of the fields of string is null, the whole formula returns empty, not the rest concatenation. Using the If-Then-Else test with the IsNull function described earlier, you can check if the area field contains a null value. If so, the formula of a blank string (set by two quotation marks sets) returns. Otherwise formula one comma, a space and then returns the name of the region. This formula can then be used with city database fields and ZIP in another formula to </result> </result> </boolean> </statements> </second> </second> </first> </more> </statement> </statement> </more> </statement> </statement> </test> </test> A city-state-zip line. If the region in the database is empty, the formula will no longer be depleted. A few factors determine whether the database will contain empty values. In many cases, this is determined by how the database is designed first. If you prefer to avoid null values appearing in the report, crystal reports allow you to convert them to a default format (normally, zeros for number fields and a blank string for string fields). To do so for all new reports in the future, check converting database NULL values to default in the Report tab of the Pull-down file options menu. You will also find the option of converting other NULL values to the default option to deal with other non-database values (such as other formulas) that may return the empty values. To set these options only for the current report, check the same options after selecting file report options from pull-down menus. Next function If {Customer.Customer Name} = Next({Customer.Customer Name}) Then {Customer.Customer Name} + continues on next page... The Next function reads a field in the next database record. This formula compares the field in the next record with the same field in the current record. If the values are the same, you know that the same client will appear in the first record on the next page, and you can note this with a text message. This formula is normally placed in the footer of the page. Note that there are no other paragraphs in this formula. Crystal report does not require another clause in the If-Then-Else formula in crystal syntax. If you leave Else off and the wrong test returns, the formula will generate an empty string. InRepeatedGroupHeader function If InRepeatedGroupHeader Then GroupName ({Customer.Customer Name}) + - continued - Else GroupName ({Customer.Customer Name}) If you place this formula in the group header of a group with the Repeat Group Header on Each New Page option turned on (this can be set when you create or change a group see Chapter 3), - continued - appears only when the group header is repeated. This also uses the GroupName function to return the group name field for a specific group. The InRepeatedGroupHeader test also comes in handy if you are resetting the variables in the formula you are inserting in the group header (assuming you will always encounter a new group when the header group is printed). Since you don't want to reset your variables in a recurring group header, you can bet your variable allocation statement on the InRepeatedGroupHeader value. You may use something like the following: If not InRepeatedGroupHeader then GroupBonus := 0 top 2 many advanced users (especially those with a programming background) often find some procedural functionality of high-level computer languages useful when designing reports. If you fall into this category (maybe you are a basic programmer), not only will the typical procedural build on the basic syntax open up increased For you, but making the same logic in crystal syntax also makes your reporting life easier. Even if you're not a programmer, you'll probably soon find that these features will be encountered manually in your more advanced reporting positions. The term Logic Build refers to the crystal features of the formula language report that enables you to go beyond the basic Logic If-Then-Else. For example, writing long duplicate formulas if-Then-Else to perform tasks such as experimenting for more than a small number of conditions, picking separate strings, or cycling through multi-value parameter fields or other arrays can be frustrating. Crystal reporting logic functions such as item selection, for loops, and performing loops make these tasks much easier. These functions enable Crystal Reports formulas to get closer and closer to a full procedural language such as Visual Basic. Select Case is very similar to its counterpart Visual Beek. It provides a much simpler and cleaner approach to testing for multiple situations and returning the right results of those complex statements if then longer can now be replaced with more readable logic and easy to maintain. See the following combination: If-Then-Else statement: If {@Ship Days} = 0 Then Shipped Same Day Else If {@Ship Days} = 1 Then Shipped in 1 Day Else Shipped in + ToText({@Ship Days},0) + Days This is a relatively simple formula that checks whether the @Ship Days formula returns a 0 or a 1, returns the value of different strings in each case. If none of these conditions are correct, a catchall Else clause displays another string value. While this particular example is not particularly complicated, it can quickly become much more difficult to interpret and maintain if more than two conditions must be tested. Choosing the case is much better suited to this kind of logic. Consider the following: Select {@Ship Days} Case 0: Shipped Same Day Case 1: Shipped in 1 Day Default: Shipped in + ToText({@Ship Days},0) + Days You may choose case from the Structure Category of the Formula Editor Operator Tree or simply typing the correct syntax. Start with the selected word looking for database field, formula, or other expression. Then, supply a few paragraph items, select each test of the phrase value (make sure you select the amount you supply to each item clause the same type of data). If you want to return the same formula to the same result for multiple different values of the selected phrase, you may separate the values after the item paragraph with commas and contain the act processor to supply a range of values. After the item clause, supply the colon (do not use semicolon this is not the end of the statement) and then supply the expression you want the formula to return if the selection value equals the item clause. Remember that all phrases derived from a Case clause must be the same type of data you cannot return one item paragraph of one string and another Return a number. After the item clauses have been defined, you may supply an optional default paragraph, followed by colon, and the phrase you want the formula to return if none of the item clauses match the Select value. Basic programmers have always enjoyed the ability to loop through pieces of program code over and over again to perform repetitive logic. This will also be useful in some reporting situations (if, for example, you need to cycle through a multi-value parameter field or repeat through the number array). Crystal reports include everywhere for loops in both syntaxes (except there is no next clause in the crystal syntax version). Loops to use a counter variable to track how many times a specified piece of logic has been cycled through. The For clause adjusts both the initial values and the end values of the counter variable. The optional step clause tells the For statement how to increase the counter variable (default is 1 if the Step clause is withdrawn). The statement is closed for by the word Do, followed by one or more statements enclosed in prantos (use a semi-clone to separate more than one statement inside pranthus). Statements inside the pronches will be executed once for each counter variable increase. The following formula displays all inputs that a user has selected in the multi-value area parameter field: NumberVar counter; Cycle through all the multi-value members//? Area parameter field for counter := 1 to count({? Region}) Step 1 Do (// build the Message variable, along with comma/space Message := Message & {? Region} [Counter] + ,); Strip off the last comma/space added by the left loop (message, length (message) - 2) first, this formula announces two variables: counter-increasing the loop for, and the message to collect the parameter field values (see the next part of the chapter for information about the use of variables). Loop for then counter cycle from 1 to number of elements in the parameter field (back by counting function). For each loop, the counter is used to retrieve the next element of the parameter field and its aggregation, along with a comma and a space, in Message. The final statement of the formula, which does not come with the ring, strips the last comma and space that was added inside the last occurrence of the loop. Note Although crystal reports now allow string formulas and variables to return up to 64K of characters (only 254 characters can be returned by string before version 9), Logic's good formula dictates adding a test in this formula that uses exit for statements to exit the loop for if the message variable may never be close to about 64,000 characters in length. If the loop tries to accumulate more than 64K characters in the variable, the run time error will occur. Making loops similar to the For loop described previously can be used to repeat statements A specific condition is met. While the For loop uses a counter variable to determine how many loops the loop runs, the While Do loop assesses a condition before each loop occurs and stops if the other condition is not correct. This structure is similar to Do and While loops used in Visual Basic and other procedural languages. The following list is a formula that adjusts a variable to a phone number database field and then uses a While Do loop to search for hyphens in the variable. As long as there's a hyphen in the variable, the

Do Loop will run a statement to pick up Hiphen and leave behind only the net numbers of the phone number. When there are no more hyphens in the variable, the While status will fail and the statement will run after the While Do loop parentheses close (variable name - which will display the number without hyphens). StringVar NewPhone := {Customer.Phone}; While Instr(NewPhone,-) > 0 Do (NewPhone := Left(NewPhone, Instr(NewPhone,-) - 1) & Right(NewPhone, Length(NewPhone) - Instr(NewPhone, -));); NewPhone uses join and split functions to block loops while the previous code is large for example loops, there is actually another built-in formula function that negates the need for variable declarations, loop logic, and delete comma/space tutorials when creating a string only containing multi-value parameter field inputs. See the following code: Regions chosen: + Join({? Region}, ,) This formula uses the Join function, similar to its Visual Basic counterpart, which takes all the elements of the array supplied in the first parameter (a multi-value parameter field actually is a array), concatenates them together, and optionally separates each with the string supplied in the second parameter. Membership does the same thing that all logic loops and variable manipulations showed earlier, with a simple function. Conversely, you may wish to get a string value or variable that contains multiple strings separated by the common delimiter (such as slash) and create an array of string values. You can create a loop that cycles through strings of one character at a time, look for delimiter (slash), and perform complex logic to extract sub-strings and add it to the array. But the Split function like its equivalent in Visual Basic will do all this logic automatically for you. Look at the following code piece (this is not a complete formula): StringVar array Regions; Regions := Split(Northwest/Southwest/Northeast/Southeast/Midwest, /) The second line of code will populate the Regions array variable with five elements by looking through the string and separating the five substrings that are separated by slashes . But, don't forget your loop capabilities just yet join and split the working function only with string values. If you have a multi-value parameter field that is set as a number, date, or other Type, you still need to use loops to extract individual elements. And if you want to create a non-string array, it may need to use loops as well, as Split only works with strings. Although this is a good example of how the While Do loop can cycle while a situation is right, it's a fairly complex process for relatively simple and alternative search function that it does. For a simpler formula, you can use the Crystal Reports Replace function, as in the following example: Replace({Customer.Phone},-,) in this case the Replace function uses three parameters: the first is the string field or the value you want to change, the second is the character or character you want to search for, and the third is the character or character you want to replace the search characters with. Note previous logic build examples are presented in crystal syntax. The logic structures of the base syntax are very similar, if they are the same, to their basic visual counterparts. Just remember that you should use at least one instance of the formula's intrinsic variable in the basic syntax to return a result to the report. Page 3 is the remaining type of formula that you may need to create boolean formula, which can return only two values, true and false. You can think of boolean formula as just test section of If-Then-Else formula. When the formula is evaluated, it will eventually return only one of the two modes. Here is the simple Boolean formula: {@Ship days} > 3 in this formula, the existing @Ship-day formula (formula number) is tested to greater than 3 (indicates the exception of the transport). It's either bigger than 3 or not! If it is greater than 3, the formula returns a real value If it is not, the formula returns a false value. When you then put this formula on your report, it will appear with boolean data type. If you're showing the field name off in file options (discussed early in the season), then you'll see the formula showing up with the correct word in the Design tab. If you format the field, you'll notice a Boolean tab in the Format Editor that lets you choose how you want real/false values to appear in the report. Although you may sometimes find Boolean formulas useful when they are actually placed in the report, you will probably use them much more as a cornerstone for other formulas. For example the Boolean formula has been shown to previously show that Xtreme Mountain Bike considers orders that took more than three days to ship as exceptions. But, Xtreme really wants to break the Shipping Exceptions Act due to last year's sales. If the customer has purchased more than $50,000 worth of goods in the past year, the three-day shipping exception will apply. However, if a customer buys less, a six-day shipping exception applies. It requires a Boolean combination formula, such as ({@Ship days} > 3 and {customer.sales last year} > 50000) or {@Ship days} > 6 that uses a combination of or Along with comparison operators, it's more complicated to create a Boolean formula. What is important to remember, though, is that the end result will still be either true or inaccurate. You can make a Boolean formula as complex as you want, using a combination of comparison operators along with and, or, not operators, but in the end, it will only result in right or wrong. Note The Pront's attention is combined around the first part of this Boolean formula. They ensure that every @Ship days are less than 3 and last year's sales exceeded $50,000 before or ing @Ship days more than 6 tests. Although this may make the formula more understandable, it is optional. Crystal's report considers all Boolean operators (and, or, rather) equally in priority arrangement (discussed earlier this season). That is to evaluate them as much as it travels from left to right from formula. There are several advantages to creating Boolean formulas in this fashion: After creating a complex Boolean formula, you can include it in other formulas as the If-Then-Else formula test section, as below: If {@Shipping exceptions} then the other transport exceptions carry within the target this makes the second formula much easier to read and understand. Using boolean formulas throughout the report, you eliminate the need to re-sample the boolean test wrapped over and over again, thereby reducing the chances of error. Even more important, you only have one formula to change if the reporting requirements change down the road. For example, if you use the @Shipping exception formula as the cornerstone of 50 other formulas in your report, and later decide to reduce last year's sales eligiby from $50,000 to $35,000, you'll only have one formula to change your report, not 50. Everyone will follow the rest of the change. You can use boolean formula in advanced record selection (covered in Chapter 8) and conditional formatting (covered in Chapter 9) to limit reports to specific records or to specific objects appearing in the report with different formatting. The crystal point of the report is the online help of a wealth of wisdom in formula concepts and built-in functions. You will find examples of every built-in function and many sample formulas that you can use as building blocks for your reports. Just click the Index tab inside help online and type in the Formula Language function or statement you want with help. Then select the function from the list to see the Help Material for that function. Page 4 As a general rule, the formulas contain your value only for the duration of a database record. If you put a formula in the details section, it will evaluate each time a new record is processed and the result will be placed in the details section. If you have a formula in a group footer, it will be evaluated when each footer group prints. In any case, the formula will not remember anything from the previous record or footer of the previous group. When the next record or footer comes along, Fully evaluates from scratch. Sometimes, though, you may need a formula to remember the material from record to record or from group to group. You may want to collect some value as the progress report so that you can print altogether in a group footer or report a footer. For example, you may want to check the value of a subtotal in a group footer. If it exceeds the specific threshold, you may want to increase the counter so that you can show how many groups report over the threshold at the end. To do this, you need to somehow save information from the record to record or from group to group. This can be done using variables. A variable is simply a holder location where Crystal reports it aside in computer memory. By progressing the report from record to record or from group to group, your formula can point to variable or change its contents. Then you can use the variable in other formulas or report its accumulated contents in a display group or footer. Note crystal syntax and basic syntax use different statements to maintain variables. Just like in microsoft visual basic, crystal base syntax requires using dim statement to declare a variable before use. And as when working in Visual Basic, you can either assign a data type to a variable when you understated it, or simply assign a value to it after you have used Dim without the data type (and the variable will automatically assign it in the data type of value you value). Because of this similarity to the basic visual, the basic syntax variables were not discussed here, as they are well documented in Visual Beek texts. The rest of the discussion of variables applies to crystal syntax. The first step in any formula that uses a variable is to declare the variable. This will set aside a certain amount of memory for the variable, based on its data type. Find the variable declarations listed in the Formula Editor's Tree Box under Variable Notices. Note that there is a different variable declaration statement for each crystal report data type. You should already consider what kind of data your variable holds, and declare the correct type of variable accordingly. If, for example, you want to track a client name from record to record, and the customer name field in the database is the type of field data, you need to declare a string variable to keep the information. You also have to give each variable a name. You can make it any descriptive name you want, provided that it doesn't start with a number, contains spaces, or conflicts with another crystal report of the reserved word language formula. You can't, for example, use variable names like Date, ToText, or UpperCase these are booked by the formula language for your built-in functions (you know whether your variable names are reserved words by looking at your color in the Crystal Report Formula Editor converting all the booked blue words). Type the variable declaration to declare the variable By variable name, such as this example: NumberVar BonusAmount; The semi-clone at the end of the statement separates this statement from the next statement in the formula (possibly a statement to assign or test variable contents). If you wish to use more than one variable in the formula, you may declare them together, again separated by semicolons. For example: NumberVar BonusAmount; StringVar BonusCustName; DateVar DateBonusReached; Your tip may be used to assign variables in other programming languages. Remember that crystal reporting is likely to behave differently to variables. You need to declare a variable in any formula you want to refer to the variable. However, even if you announce a variable and assign a value to it in a formula, and then re-declare it in the formula which appears later in the report, it will keep the value from the first formula. Unlike many other languages, declaring a variable more than once in Crystal Reports does not reset its value to zero or empty (excluding local variables, as described in the section below). These considerations apply to both Syntax, Crystal and Basic. Even if you are used to using dim statement only once in basic visual, you should use it with basic syntax in any formula where you want to refer to a variable. If the variable is declared with a Dim statement in another formula, the announcement will not reset its value again. The whole idea and benefit of variables is that they maintain their values as the report progresses from record to record or from group to group. Therefore, for variables that have real profits, they must keep their values during the reporting process. And since you may have multiple formulas that you want to point to the same variable, you should be able to point to a variable in a formula that was already declared and assigned a value in another formula. Exactly how long and where a variable holds its value is determined by variable amplitude. If a variable has a narrow range, it will retain its value only in the formula originally announced Any other formula that refers to a variable with the same name will point to a brand-new variable. If a variable has a wide range, its value will be preserved for use not only in other formulas, but also in sub-reports within the original report. (Sub-reports are covered in Chapter 13.) Below are three additional words you can put in front of your variable declarations (or use in place of dim statements in the basic syntax) to determine the variable domain. The local variable only remains in the range for the formula in which it is defined. If you declare a variable with the same name in another formula, it will not use the first formula of value. Global The variable remains in scope for the duration of the entire main report. You can declare a global variable in one formula, and another formula Unable to use the contents placed in the variable by the first formula. But global variables are not visible in sub-reports. Variable sharing not only stays in the range for the total duration of the original report but can also be referenced in formulas in sub-reports. You can use common variables to pass data around the original report, back and forth between the original report and sub-reports, and from sub-report to sub-report. Add these keywords versus variable declarations to determine your domain, as follows: Local NumberVar BonusAmount; Available to the entire original report shared DateVar DateBonusReached; Available to the main and sub-report tip if you leave out the keyword range variable in crystal syntax, the default range for a global variable will be it will be available to other formulas in the original report, but not to subreports. If you use dim statement in basic syntax, the default domain will be the local variable will only be available in the rest of the formula announced for use. If you don't want to use the default domain, make sure you always add the right domain keyword. And make sure you add keyword to the declaration in each formula that will be using the variable. After you announce a variable, it doesn't do very well if you don't assign a value to it. You may want to use it as an accumulator, to add one to it every time some conditions for the database history are met. You may want to assign a string value to it, concatenating additional string values onto the variable as records progress. Then you may display the accumulated variable value at the foot of the group, and assign a variable blank string in the group header to start the whole process again for the next group. Tip If you announce a variable but don't assign a value to it, it takes the default value based on its data type. Numerical and currency variables by default to 0, string variables by default to a blank string, Boolean variables by default to incorrect, and default date variables to date 0/0/00. Date/time and time variables have no default value. Crystal syntax provides two instances where you can assign a variable of a value: at the same time the variable is declared, or in a separate line later in the formula. In both cases, you must use the assignment acter, consisting of colons followed by an equal sign, to assign a value to a variable. It's important to make it easy to get confused and just use equal marks on its own. The equal mark only works for comparison you should put a colon in front of the equal mark so the assignment work properly unless you are using the basic syntax, in which case the equal sign itself is used for both assignment and comparison. Here's a crystal syntax example of assigning a variable value on a separate line: WhilePrintingRecords; NumberVar CustomerCount; CustomerCount := CustomerCount + 1 Here, the CustomerCount The first line (terminated with a semi-clone) is declared and assigned on the second line. In this particular formula, the CustomerCount variable will keep its value from record to record, so it will be increased by one every time the formula is executed. If you want to reset the variable value of CustomerCount in a group header, you need to reset it to 0. Here's a crystal syntax example of how to declare and assign a variable at the same time: NumberVar CustomerCount := 0; Here the variable is announced, followed by the assignment operator and the amount of allocation variable. In this example, inserting this formula into the group header will reset the CustomerCount variable at the beginning of each group. Note that a semi-clone does not have to appear in the last line of a formula, as it is used to separate one statement from another statement. If your formula only declares and assigns a variable, you will need a semi-clone at the end of the statement/assignment. You don't need to assign value to a variable every time the formula runs, and not every time you need to assign the same value. The creative use of logic structures, such as If-Then-Else or Select Case, along with variable assignment, provides the flexibility of a report that rivals many programming languages. Look at the following formula, which will announce and conditionally assign multiple variables: CurrencyVar BonusAmount; StringVar HighestCustName; DateTimeVar DateBonusReached; If {Orders.Order Amount} > BonusAmount Then (HighestCustName := {Customer.Customer Name}; DateBonusReached := {Orders.Order Date}; BonusAmount := {Orders.Order Amount}) Look at this formula closely. Assuming it is placed in the details section, it keeps track of the highest order amount as records progress. When an order exceeded the previous high value, the client who placed the order and the order date was added to the variables. Then the new high order amount is allocated to the bonus amount. Below are some important references to the note about formulas: Multiple variable assignments separated by semicolons inside the pronth. They will all run, but only the last statement will determine how the formula will appear on the report. In this example, the latest statement uses the currency data type, so the formula will appear on the report as currency. If you are tracking bonus sums, dates, and customer names for a specific group, such as a region or country, be sure to reset the variables in the group header. If it fails to reset variables, and the next group does not have an order as high as the value in the previous group, the previous group values will also appear for the following group. If you want to track quotas or similar values for both groups and reporting levels (e.g., you want to see customer rewards for each area and for the whole report), you need to assign and maintain two sets of variables: one for The group level is reset in the group header and one for the reporting level is not reset. In the previous example, you saw how to sum values in variables in the Details section, and how to reset them by assigning 0 value in group header (or in another area of report). You also need a way to show exactly what is available in a variable in the report, or to use the variable value in the formula in some other way. To show the contents of a variable, you simply need to declare it. If the formula contains no other statement, the variable announcement will also return it as the formula value. For example, you may place the following formula at the foot of the group to show the client that it has reached the bonus in the region group: StringVar HighestCustName you neither need to insert any other statements in the formula to show the variable value nor even need semicolon at the end of the declaration line it is the last line in the formula. There may be conditions that you want to show the contents of a variable but you will use other statements to assign the variable in the formula. In that case, just the variable announcement will not display it, as the declaration statement will not be the last line in the formula. In this situation, just add the variable name as the last line of the formula. Then this work will display the variable contents when the formula runs out. Here's an example: CurrencyVar BonusAmount; StringVar HighestCustName; DateTimeVar DateBonusReached; If {Orders.Order Amount} > BonusAmount Then (HighestCustName := {Customer.Customer Name}; DateBonusReached := {Orders.Order Date}; BonusAmount := {Orders.Order Amount}); HighestCustName This formula performs test and variable assignments as before, but the last line of the formula simply shows the HighestCustName variable which is a string variable. Therefore, this formula shows up with small x s in the Design tab (if displaying the field name in file options is turned off), and the variable contents of HighestCustName will be shown whenever the formula runs out. You can even step ahead by testing and assigning variables and then using them later in other calculations or concatenations. Here's another fit of this formula: CurrencyVar BonusAmount; StringVar HighestCustName; DateTimeVar DateBonusReached; If {Orders.Order Amount} > BonusAmount Then (HighestCustName := {Customer.Customer Name}; DateBonusReached := {Orders.Order Date}; BonusAmount := {Orders.Order Amount}); As of this order, the amount to beat is & ToText(BonusAmount) + set by & HighestCustName & on & ToText(DateBonusReached,M/d/yy) This formula not only announceds variables, it also conditionally assigns them and then concatenates and displays them, converting them to text as necessary. As you may have been collected by this time, formulas that contain variables are often affected where they are physically placed on the report. Check values and assign variables if Processing the record to record, you need to insert the formula into the details section. If you want to show the accumulated sums for each group, we will place a formula in the footer of the group to show the sum of variables. To reset the variables for the next group, a formula that resets them in the group header must be inserted. However, just inserting formulas in these sections does not necessarily guarantee that they will actually evaluate the report in that section or during the logical formatting process (during which it prints a group header, then prints detail sections for that group, then the footer of the group is printed and the like). Consider the example below. Figure 5-4 contains a report that calculates a running total using a variable. The variable accumulates order values in each detail section with the progress of the report. Figure 5-4: Executing the whole using a variable as you can see, reporting a detailed report is simple, there is no group. Total running stacking orders as the report progresses. The formula includes the following variable assignment: CurrencyVar MonthlyTotal := MonthlyTotal + {Orders.Order Amount} In Figure 5-5, the report is grouped by Order Date, using for each month grouping. In these situations the desire to reset the whole running for each month, as the viewer evaluates each month on its own. Accordingly, the sum of the variable running MonthlyTotal should be reset in each group header with the following formula: CurrencyVar MonthlyTotal := 0 Fig 5-5: The formula in the group header does not work to reset running the whole however, even when this formula is placed in the report group header, the desired result will not be achieved. Note that not only does the sum running from the group header not reset to zero, it also shows a smaller value than the previous group. Why add a group leading to weirdness with the whole running? This happens because the total collection formula running at a different time than when it actually displays it on the report, and because the formula for resetting the whole running is being evaluated at another time during the reporting processing. The formula for total collection running is being calculated while the records are reading from the database, not when the records are grouped and are actually being printed or formatted. Also, the formula that resets the running sum is actually processed only once, at the very beginning of the reporting processing, not when each group header is printed. These formulas are said to calculate in different reporting passes compared to the pass that actually formats the report. Therefore, the total running is now calculated for each record before crystal records sorting reports are placed in the group. In addition, the whole running is actually reset to zero only once before anything else happens in the report. Crystal's report generally breaks its reporting processing to the following three passes, during which certain types of Automatically evaluate. Before reading records occur before any records are read from the database. If formulas do not contain any reference to database fields or summary functions, they will be calculated in this pass. These formulas are sometimes called flat formulas. While reading records occurs as records are being read from the database, it is done before any record selection, sorting, or grouping. Formulas that contain references to database fields but do not contain any subtotal or summary functions are calculated in this pass. These formulas are often called first pass formulas. While printing records occur after records have been read and are being formatted to display or print. Sorting and grouping occurs during this pass. Formulas containing sum, average, or other summary functions are included in this pass. These formulas are often called second pass formulas. In most cases, you can report crystal trust carefully determining where its passage requires evaluating the formula. But the stark exception is when a formula uses variables. If a formula simply declares a variable, or declares the variable and assigns it a literal or fixed value, crystal reports will evaluate which formula passes in before reading records because it makes no mention of database fields or summary functions. If you have assigned a variable to a database value, it will evaluate the crystal reports that the formula passes while reading records (the formula will become a first passing formula). Only if you have some summary type or subtotal function in the crystal reporting formula will automatically evaluate the formula while printing pass records (the formula then converts to a second pass formula). This default behavior can produce very strange results, as the previous total running example shows. The formula for total running collection makes reference to the order of the database field value and thus evaluates on the first pass (WhileReadingRecords). This whole collection was grouped running just fine before the report. However, when the report was grouped by Order Date, records appeared on the report in a different order than those read from the database, resulting in running totals no longer appearing in logical order. And to further complicate the issues even further, the formula that resets the whole running variable makes no mention of the database fields, thus becoming a flat formula (BeforeReadingRecords) and only once at the beginning of the reporting processing, rather than being processed at the beginning of each group. When you use variables in a formula, you may need to force the formula to evaluate in a different pass than it is by default. You will get it done by changing the formula evaluation time. To do this, add an assessment-time statement as the first statement in the formula. Look at the Tree Function Formula Editor box and you'll notice the Evaluation Time section. Open that section to see several assessment-time statements Now it has to be more self-explaning. To compel a formula that accumulates the running total to the second pass, where it will calculate the total running correctly after the records have been grouped, add the WhilePrintingRecords assessment-time statement to the formula, as follows: WhilePrintingRecords; CurrencyVar MonthlyTotal := MonthlyTotal + {Orders.Order} Caution Don't get confused if you can insert t insert a subtotal subtotal. summary, or grand total on a second pass formula. When you click on this type of formula in the Details section, no subset, summary, or large total options will be available in pull-down or pop-up menus, because subtotals, summaries, and large totals are calculated in the WhilePrintingRecords crossing. If the formula is already evaluated in that pass, you can create a large summary or whole on it. Now, to ensure that the formula that is resetting the whole running actually happens when groups are formatting, rather than once only at the beginning of the report, it will have to pass WhilePrintingRecords as well. WhilePrintingRecords; CurrencyVar MonthlyTotal := 0 is an evaluation-when it may be self-explanative, the assessment then takes one argument: the name of another formula. This will force a formula to evaluate after another formula when evaluated in the same pass and are in the same report section. Since crystal reports automatically evaluate formulas that contain other formulas in proper order, you will use this function very rarely. However, it may need to be used with formulas that contain variables. When crystal reports evaluate two formulas that contain a variable in the same reporting section, the order in which they will be evaluated is not predictable. One example is if it has two formulas in a group footer. The first formula shows the values of variables (assuming those values are set in other formulas in the details section): WhilePrintingRecords; CurrencyVar BonusAmount; StringVar HighestCustName; DateTimeVar DateBonusReached; The highest order + ToText (BonusAmount) + by + HighestCustName + on + ToText (DateBonusReached,M/d/yy) the second resets the variables to zero or an empty string to prepare for the next group: WhilePrintingRecords; CurrencyVar BonusAmount := 0; StringVar HighestCustName := ; DateTimeVar DateBonusReached := DateTime(0,0,0); Since it is likely that the formula that resets the variables will be evaluated before the formula that shows them, you have two choices. First, and possibly most logically, simply move the formula that resets the variables to the group header. In this way, variables will reset when a new group begins after they are displayed at the foot of the previous group. Or if there is a logical reason why both formulas should exist at the foot of the group, you can use EvaluateAfter in a formula that resets variables as follows: EvaluateAfter ({@Bonus CurrencyVar BonusAmount := 0; StringVar HighestCustName := ; DateTimeVar DateBonusReached := DateTime(0,0,0); By inserting the evaluation then as the first statement in the formula, you will force the reset formula to be evaluated after the display formula. Because you have to evaluate this formula after a formula that exists in the second pass, there is no need to include WhilePrintingRecords in this formula. Tip As you start adding formulas that calculate and reset variables, you may find quite a few examples of things that appear in detail and group header sections that show zero or other unnecessary information. You can't remove formulas from these sections because then they don't evaluate correctly. To hide them, just click Suppression on the Format Editor shared tab. Then you'll see them in the Design tab, but not on the Preview tab or any other report output. It's fairly common to learn how to use some spiffy features from a tool, and then to use them to too! Variables have this potential. Although they are fast and if used by judgment, do not consume significant memory or additional resources, they can sometimes take too many pills. If you find use for variables, first look closely at your report to see if there is an easier and faster way to accomplish the same thing. Figure 5-6 is an example of a report that counts orders that exceed the $1,000 bonus level. The number of orders must be shown both at the group level and at the end of the report. Figure 5-6: More than $1,000 bonus reports using variables to accomplish this need to create multiple formulas. Two variables are also required: one to collect the number of bonus orders for each group, and one for the total number of reports. The following formulas are. @Bonus Calc is placed in the details section and suppressed: WhilePrintingRecords; NumberVar CountCustomer; NumberVar CountReport; If {Orders.Order Amount} > 1000 Then (CountCustomer := CountCustomer + 1; CountReport := CountReport +1) @Show Group Bonus is placed in the footer group: WhilePrintingRecords; NumberVar CountCustomer; This customer had + ToText(CountCustomer,0) + bonus orders. @Reset Group Bonus is placed in the group header and suppressed: WhilePrintingRecords; NumberVar CountCustomer := 0; @Show the report is placed in the report's footnote: WhilePrintingRecords; NumberVar CountReport; This report had + ToText(CountReport,0) + bonus orders. While this will work, there is a much easier way to accomplish the same job with only one formula using no variable. Create a single formula, put it in the details section and suppress it. It simply is composed of the following: If {Orders.Order Amount} > 1000 then 1 when you put this in the details section, it fixed the number 1 when an order returned more than $1,000. If the command is below $1,000, formula no. 0 returns (because the formula is numerical and there is no other paragraph, if the if test fails, 0 returns). After that you simply need to insert subtotal and report large totals in the formula to calculate the group and the total report. The result: the same sums with much less effort. This simple technique of assigning a value formula 1 if a pass test can become the cornerstone of many report type statistics you may have to write. You may also be able to save time by running entire fields instead of formulas with variables. The total report is running earlier in the season that suggests the evaluation time is a perfect example. In this type of report, there is no need to create formulas to calculate the total running. Running all fields will be covered later this season. The point of many types of illustrated formulas in this chapter is included in the sample report in this book along with the website. Look at www.CrystalBook.com formulas, RPT and rewards. RPT to see how these, and similar formulas, are implemented. Page 5 Crystal is reportedly designed as a developmentable reporting tool. According to the formulas, this means that you can develop your functions to add to the Tree Function box if the crystal report is not the one you need. In Crystal 9's report and later, this feature is enhanced with custom functions that you can create directly in your report, or add to the repository to be shared with other crystal reporting users (custom functions are covered in Chapter 6). However, crystal versions report before 9 custom feature functions. Also, in some cases, custom functions may still not provide sufficient functionality for your specific business needs. For these conditions, you still need to create your own functions that appear in the Formula Editor. Look at an example of built-in functions that appear under the category of additional functions. The functions in this category are really built. These functions are supplied to crystal reports by user function libraries (UFLs). The UFL is supplied to Crystal Reports by an external dynamic link library developed in another programming language. You can write down your custom functions using a Windows programming language, such as C++ or Visual Basic, and pop them up in this section of the Function Tree box. For example, you can write a function that will calculate the number of business days between two dates, excluding any weekends and companies that are available in an external database. Or you might write a UFL that reads a value from an external piece of proprietary measuring equipment and supplies value to your report. The file name that supplies the UFL will appear as a category in the list of additional functions you may click the plus sign next to the file name to see the existing functions supplied by that file. As with other functions, double-click on the function name and supply the necessary arguments within your formula. Tip Although the need for external UFLs is likely to be greatly reduced by crystals reporting custom performance and tank You may still need to use them to connect to external databases, external equipment, or any other functionality that has not been provided by crystal formula language reports (which is what Crystal reports custom use functions). Information about creating user functions libraries with basic visual can be found on the book's website. Look at www.CrystalBook.com page in certain situations, it is inevitable to use formulas with variables (discussed in the previous chapter). However, many of the examples shown in the past can actually be done without even creating a formula. If you need to collect, display, and reset running totals, you'd probably prefer running the entire field. A whole running field can be inserted just like a database field. This gives you great flexibility to collect or increase value as the report progresses, without the need for formulas or variables. Figure 5-7 shows the Report N High (discussed in Chapter 3) that shows regional subtotals for the top five regions in the United States. This particular report above N does not include others. As mentioned in Chapter 3, this makes the great total report not agree with the total of the entire group. Large sums are based on all reporting records, not only those falling into the top five groups. Using running entire fields is the perfect answer to this problem. Figure 5-7: A big total problem with the high N report and no group of others all new running the entire fields of Field Explorer created. First, select the Category of Running Total Fields in Field Explorer. Click the new button in the Field Explorer toolbar or right-click the Running Total Fields category and select New from the pop-up menu. You may also select an existing field in the Details section, right-click and select Insert Running Total from the pop-up menu. Create whole running field dialog box will appear as shown in Figure 5-8. Fig 5-8: Create the Running Whole Field dialog box starting by giving running in the whole field of a name (if you don't like the default name given by crystal report). Can contain characters and mixed spaces and will not conflict with formula field names or databases. Report crystals before running the entire field name with pound sign (#). If you select a detailed field and right click method to insert the whole running field, the field you choose will now appear in the field to sum up the drop down list, and a default summary function will appear in the Summary type drop down list. If you are creating a new field running whole of Field Explorer, Report, Database, or Formula Field that you want to calculate the whole running by selecting the field in the list of tables and fields available and clicking the right arrow next to the Field to select summary box. Select the calculation type that you want to use the pull-down summary list type. If you just want to increase the total running by one for specific records, use count or DistinctCount (Depending on how unique the field you summarize) comes with any field of report that does not contain empty values. The emptys do not increase. Other functions available altogether are running the same as those available for summaries. A look at Chapter 3 is available for detailed description of the summary. Select when you want the whole running to increase, with the selection in the evaluation section. Then, select when you want the whole running reset, with the selection in the Reset section. If you select a field in the list of existing tables and fields then click the arrow next to the Radio Shift field button, the whole running increase or reset every time a new value appears in that field. If you click the Change Group Radio button, then you can select an existing report group in the pull-down list. The total running increases or resets every time the select group changes. If you click the Use a Formula radio button, then you can click the formula button next to it. The formula editor will appear, where you can enter a Boolean formula that triggers when running the whole field is enhanced or reset. When you have completed the Create running dialog box in the whole field, click OK. Total running now appears in the Explorer field and can be dragged and dropped in the report just like a database field. If you want to edit, rename, or delete the entire running field, you will find these selections in the Explorer field. You can also right-click on a whole field running on either design or preview tab, and select Edit Field Object from the pop-up menu. To solve the problem with the top N report without others, simply create two whole running fields: one to calculate the number of customers: and one to calculate large total sales: the location of this total running in the report footer rather than the large sum of fields from the details section. Since running total evaluations only during while printing passing records, additional records in the group of others will be included in the report's footnote. Figure 5-9 indicates the correct sum is now displayed in the above N report. Figure 5-9: Correct above N report using running total caution because running entire fields while printing records passing the report is calculated, you can create running whole fields based on the second pass formulas. Page 7 One of the most frequently heard questions from crystal reporting users is how do I share my formula with other reports or with other crystal reporting users? Often, too, a reporting designer will find a set of Logic formulas that are specific to their type of business or type of report that should be used repeatedly in many other formulas. After that question, is there any way to reduce the need to type these same parts of formula over and over again? In previous versions of the Crystal Report, the solutions to these requirements were typically not very subtle. Some users opened two reports Reduce the size of the windows inside the crystal report designer, and drag the formula from one report to another. Others opened a report, edited a formula, copied the contents to the clipboard, and snedented the contents on a new formula. Or a user may copy the contents of a formula in a report, close the first report, open the second report, create a new formula and paste the formula text from the first report in the Formula Editor. Still others keep their library of formulas in a text file or word processing document, cut and paste between different formulas in different reports and this document regularly. Now, Crystal Reports makes this much easier by providing the ability to create its own reusable functions, called custom functions. Not only can custom functions be used repeatedly in different formulas in the same report, but they can be added to the crystal reporting repository for use in other reports and for use by other crystal reporting users. Page 8 custom function is very similar to a regular function you already report in the Crystal Function Tree Formula Editor (read more about general formulas and function tree in Chapter 5). For example, the Built-in ToText function in the formula language of crystal reports from, for example, will convert a numeric data type to text data type. The built-in left function returns a certain number of characters from the left of a string value. And the built-in Round function will move a numeric value away to a certain number of tening places. But what if you have a certain requirement in your company to use a function that is not included as part of the Built-in Crystal Report functions? You may need to calculate for example the number of business days between two dates, excluding weekends and companies. Or you may want to calculate a discount for customers in lots of formulas. However, you want to base this discount on a number of factors, including the size of the order amount, the number of orders the customer placed last year, and how long the customer has been. While you can probably use crystal reporting extensive formula language to create these specific formulas, use Logic repeatedly in more than one formula, or share Logic with other reporting designers, where a custom function becomes really useful. The existing custom functions (either that you made after creating the report, or added from the repository) appear in the editor formula just like the built-in functions. However, custom functions appear in your custom functions area of the function tree, as shown in Figure 6-1. Figure 6-1: A custom function used in a custom Formula A function can accept parameters or arguments R, just like a regular function (R arguments are discussed later in the chapter in more detail). And as with regular functions, you can use custom function over and over again in many formulas you want to . If the main logic changes your custom performance (maybe business holidays or percentage discount row changes with new year), changing the custom function in one place will reflect itself in all formulas that use that function you have to edit any formula to change the formula logic. Top 9 if you install Crystal Enterprise 10, ship your tank with a starter set of custom functions. However, you will probably soon discover a use for yourself. Creating a custom function is very similar to creating a crystal reporting formula of many common built-in functions and operators are otherwise available in regular formulas for your use in a custom function (Chapter 5 covers creating formulas in more detail). There are two ways to create a new custom function: base it on an existing formula, or create it from scratch. If you have a formula in a report that contains the original Logic set that you wish to use for your custom function, you may base your custom function on it by extracting the custom function from the formula. Do this by doing these steps: set up the Formula Workshop by clicking the Formula Workshop button in the Expert Toolbar, or selecting the Workshop Formula Report from pull-down menus. In the Formula Workshop tree, right-click on the custom report functions category and select New from the pop-up menu. You may also just click on the Custom Report functions category to select it and click the new button in the Formula Workshop Toolbar, or click the down arrow next to the new toolbar button and select custom function from the drop down list. Type in a name for your custom function. Select a name on the naming restrictions discussed later this chapter. Click the Use extractor button to base your custom function on the existing report formula. The custom extract function will appear from the Formula Bar dialog box as shown in Figure 6-2. Figure 6-2: Create a custom function based on a formula creating any desired changes to R arguments that will be replaced for database fields in the original formula. You may also add descriptive text to the function definition that will appear when you use the function in the expert formula (discussed in Chapter 5). If you want to change the original formula to use function, check the change formula to use the new Custom Function check box. To offer optional items, such as default argument values and other descriptive texts, click the Enter More Info button. Click OK to save the new custom function. Your custom performance naming restrictions are more limited in

choosing names for your custom functions than you are for formulas. Since custom functions appear in the Formula Editor function tree next to all built-in functions, there are several limitations: the custom function name should begin with a letter you cannot start a function name with a number. Also, you can only include letters, numbers, and characters emphasized in The custom function name of other special characters (such as %or # marks) cannot be included. Custom function names cannot contain spaces you can only use emphasis to separate a custom multi-word function name. However, you can use the upper and lowercase mixed letters to help you determine the custom multi-word function name. You cannot use a name that is currently made by a crystal reporting function. For example, if you try to create a custom function called ToText or Left, you will get the error message. You may notice that the custom function names supplied by Crystal contain a two-character CD default. Although adding your draft is definitely not required, it may be a handy way to identify a specific group of custom functions. The prediction of custom function names may also help prevent naming conflicts with other custom functions in the repository, as well as conflicting with the internal function names. One of the first settings you need to create when creating custom functions, as opposed to formulas, is handling the argument function. R is the parametric argument you create within your function to accept a value from the formula it calls it. It then uses this value (possibly along with other arguments that are supplied) to perform some calculation or logic and returns an result to the recall formula. If you think of crystal reports made in functions, most need that you supply arguments when you use them in formulas. For example, if you are using the ToText function to convert a number, date, or other type of non-string data to a string, you need to supply at least one argument R: the non-string value to be converted. And the left function, which returns a certain number of characters from the left of a larger string, requires you to supply two arguments: string to retrieve a subset of characters from, and the number of characters to retrieve. In cases of internal functions, the arguments you supply must match specific data types. ToText's single-argumentative example requires a string argument to be presented. The left function requires a string argument R and an argument R of numbers. When you create your own custom functions (either by extracting Logic from the existing formula or creating the function from scratch), you should consider if your function needs to accept the argument of the recall formula, how many arguments the function you need, what kind of data the argument will be, and if you want to supply one or more default values for each argument. When extracting a custom function from a formula, such as the one shown in Figure 6-2, the number and types of data in your function are automatically determined by the number of unique database fields included in the original formula. In the example shown in the figure, the main formula of the custom function is based on two database fields: {Orders.Order Amount} and {Customer.Last Year s Sales}. When the custom function is extracted from this formula, Fields are removed from the function and replaced with two arguments, both of which require currency data types (data types from the main database fields) to be supplied. The argums that identify the extraction process are listed in the List of Custom Extract function argoms from the Formula Dialog box. You will see the main database fields that are replaced by argument R, the type of argument R data (both of these columns are for your information and cannot be changed), the name of the argument R in the custom function, and the description of the argument R. By default, the extraction process will be numbered sequentially from number 1, before the R number of the argument with the letter v. If you simply wish to allow the default v argument number to remain, you may ignore this part of the custom extract function from the Formula Bar dialog box. However, in that case, the function shown in Figure 6-2 like this appears in the Formula Editor function tree: CurrentYearDiscount(v1, v2) It is more likely that you want the argum in your custom function to have more meaningful names, perhaps replace v1 with OrderAmount, and v2 with PreviousSaleAmount. To do so, rename the argom names v1 and v2 in the dialog box. You can optionally add comments to any arguments that will appear in the Formula Expert (described in Chapter 5). When you extract a custom function from a formula, you will notice the More Info button in the Custom Extract function from the Formulas dialog box. If you click this button, the Custom Function Properties dialog box will appear. The shaded areas of this dialog box just to reference them cannot be changed here. Summarize and explain interchangeable arguments with those in the custom extract function from the Formula Change Function description dialog box and change the argument in both places it in another. However, the category, author, display in the expert review box, the default argument values, and the help text can be changed only in this dialog box. If you look at custom functions supplied with Crystal Enterprise's 10 default repositories, you'll notice that they are classified in a multi-level deep hierarchy (e.g., the custom cdDateDiffSkipHolidays function in the history category, which is in the crystal category). You may set up categories and hierarchies for your custom functions. Do this by specifying the category in the Category textbox. If you want to create a hierarchy for categories, separate the category names with slashes. For example, inserting a function in the Sales/Orders category will show a sales entry in the Formula Workshop tree with plus sign. When you click the Plus sign, the order subset will appear with the Custom Function currentYearDiscount. This is, in fact, an opinion field that you can either leave blank or fill in with the text of your choice (possibly the name of the person who developed Custom function). This value only appears when you edit the custom function or use it in the Expert formula. This text will not be available in the formula editor. This check box will determine whether the custom function will appear in the list of functions available when using the expert formula (covered in Chapter 5). If you check this box, then you will see the custom function in the Formula Editor function tree, but not in the list of existing custom functions when using the expert formula. For specific arguments, such as those that may accept section codes, months of year, or other common values (the order quantity/sales samples of the previous period here probably won't fall into this category), you may wish to supply a list of one or more default values that a user can choose from when they use the custom function in the expert formula. To supply these, click the default values box for the argument you want to set defaults for. This will show the default values dialog box. Here, you may type in a default value that you want a user to be able to choose; Then click the Add button. This will add the value to a list in the lower part of the dialog box. If you want to add extra values, type them in it and click the Add button. If you want to remove the value already added, select it in the lower list and click the Delete button. And if you want to change the order of the default values you added before, select the value in the list that you want to move and click the up or down arrow to the right of the list. When you click OK, the default value list will be added to the Custom function, and it will appear in the reasoning list in the Main Custom Function Properties dialog box. Clicking the Help Text button will simply display another dialog box where you can type in the free form text description custom function, provide more details on its arguments or return value, or other useful information. This help text will only be visible when you edit this function later, or use it in the Expert formula. This text will not be available in the formula editor. Caution there are certain requirements the existing reporting formula should be used as the basis for a custom function. For example (and this list is not all inclusive), the formula cannot contain any evaluation time functions (such as WhileRecords), any summary functions used with database fields (such as Sum({Orders.Order Amount}), or any variable that is domained other than Local. If you attempt to extract a custom function from such formulas, you will either receive an error message indicating that the formula cannot be used, and explaining why, or a function that may not be the way you expected it to do. You may either copy the formula to clipboar d and paste it into the Custom Function Editor when creating the function from scratch (make changes to the copied formula allows it to work as a function), or edit the report formula Explicitly adjust the variable range to local), and then extract the function from the updated formula. There may be times when an existing report formula is not available as the foundation of your custom function. Also, you may need a custom function that is more complex than an existing formula, or the existing formula may contain elements that prohibit it from being used as the basis for a custom function (it may use global variables, evaluation time, or other restrictive features). In these cases, you want to create a custom function from scratch using the Custom Function Editor. In this case, the steps to create the custom function are the same. Create and name the new custom function in the formula workshop as described earlier in the chapter. However, in the Custom Function Name dialog box, click the Use Editor button to display the Custom Function Editor inside the Formula Workshop as shown in Figure 6-3. Figure 6-3: The Custom Performance Editor custom performance editor is roughly identical to the formula editor discussed in detail in Chapter 5. The difference is that the field tree, which contains a list of database fields, is not visible. And the function tree, which contains a list of crystal-built reporting functions, contains a slightly diminished set of internal functions. This variance is because custom functions are designed to be independent of any particular report they are placed in; You also have a limited set of internal functions (such as evaluation time, printing state, and others) because of the stateless nature of a custom function is prevented. Tip Even if the function tree contains a dropped set of options, it still displays a custom function category where you can choose another existing custom function to use in the current custom function. You may now create only logic custom function with two built-in clicking functions in the function tree and operators in the operator tree. Of course, you may type in these cases as well as the related formula code. And as with the Formula Editor, you can check your custom performance syntax before saving by clicking the Check button on the toolbar or typing ALT-C. When you create a custom function, you need to select either crystal reporting formula language (or syntax) just like you report when creating a formula. Select the syntax you want from the drop down list to the right of the Custom Function Editor toolbar. The function tree and the action tree will be adjusted based on the selected syntax. You will also see little difference in the formula text that is automatically added by crystal report. The basic assumption of creating a custom function directly in the Custom Function Editor is that the design of a formula has passed to accept any necessary values Function of the calling formula; If you've created functions in a programming language like Visual Basic, then this should be a fairly up-to-be process. If you've planned on a programming language in the past, then getting the hang of custom function creation may take some time. However, if you are familiar with crystal reporting formulas in general, you should be able to use your existing knowledge to create custom performance very quickly. Here is an example of the basic syntax custom function that returns the spelling name of a company section based on the shortened name moved in as the argument: the SpelledDepartment function (abbreviated string) as the selection string item stands for HR SpelledDepartment = HR SpelledDepartment = HR Spelled Case IT Spelled Department = Information Technology Case EXEC SpelledDepartment = Executive Case ENG SpelledDepartment = Engineering Case Else SpelledDepartment = Abbreviation End Select End Function The basic function (no pun intended) layout of a Basic syntax function is formula code within function and end block function. The function statement declares the function name (the same name you used when creating the function), a list of each argom and the type of data the function must accept (a string argom named abbreviation in this case), and the data type that the function returns to the recall formula (the data type is optional as part if you do not include it, the return data type function will be assigned by your data type to the name. The next function is determined within the function). The last statement within the block function of the end function assigning a value to the function name will determine what the function will return to the recall formula. In this case, the function uses a Select Case (operating tree control structures) to test the different values of the passing acronym argym and adjusts the function name to a literal string based on the argyle value. Here's the same function in Crystal Syntax: Function (abbreviation StringVar) Select abbreviation for HUMAN RESOURCE CASE: HUMAN RESOURCES CASE IT: Technolgoy Information Exec Case: Executable Case ENG: Default Engineering: Abbreviation in this case, there is no function end block function all formula text included in the implicit function is considered inside this block. The passing argom to function is included in the pronths immediately after the function keyword, before the data type that the argom is assigned (the data type keywords are the same as those for declaring variables see the variable declarations section of the operator tree for proper spelling). Similar to the basic syntax example, the last statement within the function will determine what function To the call formula. In this example, the Case statement, which matches the passing argument R, returns a string value to the recall formula. In previous examples, the custom function accepted a simple unit string value as argument R and returned a simple unit string value as a result. However, you may have more complex custom functions that need to deal with more than one argument R, or a variety of complex data, such as arrays and range values. While using the Extract function, discussed earlier in the season, can create custom functions with multiple arguments, it cannot create functions that accept or return a variety of complex data. To accept range values or arrays as argest, or move the same data types back to the formula, you need to use the Custom Function Editor. You may also create a specific type of argument in a custom performance editor called optional reasoning. By placing the optional keyword against an argument R name, you specify that the function calling formula doesn't have to supply this argument R. If the formula supplies the argument, it will cancel the default value for the argument you should supply if you declare an argument as optional. For example, if you create the following as the first line of a basic syntax function: Function DaysBetweenDates_ (BDate As Date, Optional EDate As Date = CurrentDate) you will be able to call the function in a formula using one or two arguments, because the second argument R is optional. If you call the function argument using two Rs, both are supplied to the function. If you call the function using only an argument R, that argument R will be supplied as R argument BDate, and the current date of the computer system clock (CurrentDate is an internal function) will be supplied as the second argument R. Adding optional argils will cause multiple function occurrences in the function tree to show the ability to invoke the function with multiple argils. For example, the custom function of the created unit with the function statement shown will appear twice in the function tree before. R Optional arguments in crystal syntax are also declared optional with the keyword, but by spelling the data type standard crystal syntax and equal colon operator is used to assign the default value. The function (DateVar BDate, optional DateVar EDate := CurrentDate) The more subtle point of points on crystal syntax and the basic when used in custom functions, such as how to declare complex data types in arguments and syntax differences between custom functions and reporting formulas, can be found in Crystal Report help online. Search for custom basic syntax functions, or custom crystal syntax functions. When you create a custom function, you may wish to change it later. One of the advantages of custom functions over copying and pasting the same Logic formula repeatedly in multiple formulas is that all formulas based on a custom function automatically reflect the change made in a custom one. In this way, a lot of effort can be removed by sharing custom functions in the repository and making a single change to the repository. The change will automatically be reflected in all formulas in all reports that use that function. Save custom functions in the repository later this season and are covered in Chapter VII. If you had added the custom function to only one report, you may change a custom function in the same way that you change the formula: select it in the formula workshop, and make changes directly to the formula text in the Custom Function Editor. When you save the changes, each formula using the function will immediately reflect the change. Note even if you are initially able to extract a custom function from a formula using the Dialog box, you need to make changes to the Custom Function in the Custom Function Editor. Once you have saved any custom function, even by extracting it from a formula, the edit should be done in the Custom Function Editor. If you have added custom functions from the repository to your report, you will notice that the Custom Function Editor Text Area is disabled for editing when you select the custom function in the Formula Workshop tree. This is because you have not disconnected the custom function from the repository. To change the contents of a repository-based function, you need to cut the function this is similar to checking the function from the repository. To disconnect a custom function from the repository, right click the function you want to edit in the Formula Workshop tree (this should be a function in the Custom Report function category you cannot cut or edit a custom function that was not first added to the report). Select Disconnect with repository from the pop-up menu. You will notice two changes: First, the Custom function can now be edited in the Custom Function Editor, and second, the small icon that appears next to the custom function name in the Formula Workshop tree will no longer display the small linked icon. You may not only edit the formula text of an existing custom function, but also edit the summary, author, category, and other nonformola items you specified for the first time when creating the function. With the function you want to edit displayed in the Custom Function Editor, click the Show Shift Properties button in the Formula Workshop toolbar, or right-click the function name in the Formula Workshop tree and select View Properties from the pop-up menu. The Custom Function Editor will be replaced with a dialog box that summarizes all other properties that can be set up for the function. Make any necessary changes to this screen and click on the same toolbar button or use the same pop-up menu selection back to custom performance editor. Once you've made changes to custom performance and saved the changes, you may want to add your new custom function to the repository to share with reports or other users. If the function is custom The modified version of an existing repository function, you probably want to reconnect the custom function to the repository with changes. The following section describes how to add a custom function to the repository. Top 10 is one of the main benefits of Crystal Reporting Custom Functions's ability to share them not only among the reports you may design, but among reports that other crystals report design users as well. This is done by saving custom performance or functions you wish to share in the repository. And if you have a custom function disconnected from the repository to change it, you probably want to save the updated function back to the repository. Steps to add a new custom function to the repository first, or to update a custom function that you cut, are the same, and there are actually three different ways to accomplish the same thing: right-click the custom function you wish to add to the repository in the Formula Workshop tree. Select Add to Repository from the pop-up menu. If you have already entered your Crystal Enterprise 10 system to access the repository, you get to do so. If the function already exists (you cut it earlier), you get to confirm that you want to rewrite the existing repository function. Select the custom function you wish to add to the repository in the Formula Workshop tree. Click the Add to Repository button in the Formula Workshop toolbar. If necessary, you will be asked to enter Crystal Enterprise. If the function already exists (you cut it earlier), you get to confirm that you want to rewrite the existing repository function. Simply drag the custom function you wish to add to the repository from the custom report functions of the Formula Workshop tree category functions to the Custom Repository Functions category. This will automatically add the function to the selected repository. If necessary, you will be asked to enter Crystal Enterprise. If the function already exists (you cut it earlier), you get to confirm that you want to rewrite the existing repository function. Your caution may be able to add a custom function to the repository or a new custom function you created yourself, or a custom function you cut from the repository to edit. Depending on the crystal enterprise system you are connected to, and your rights have been granted, you may be able to add or update custom functions. If you receive an error message while trying to add a custom function to the repository, contact your Crystal Enterprise manager to confirm that you have the right to change the repository. Page 11 Is easy when you have created a custom function, either for use only in current report, or shared from repository, using function in formula. You may create a formula that makes use of function with expert formula (covered in details 5). Or if you create a formula with formula editor, the custom functions you created in the current report will appear in the Custom Tree Functions category function. Custom functions in the repository, however, do not automatically appear in the function tree unless you add them to the report first. To add a custom repository function to the report, expand the custom repository functions of the Formula Workshop category until you find the custom function that you wish to use (you may be asked to enter crystal enterprise first). Select the function you want to add, and click the Add Report Toolbar button in the Formula Workshop toolbar. You can also right-click the Name function in the Formula Workshop tree and select Add to Report from the pop-up menu. And finally, you may simply drag the custom function you wish to add to the report from the Custom Repository functions category functions to report custom tree category formula workshops. If you choose the custom function to add to the call report other custom functions as well (remember, one custom function can call another), you will be notified that additional custom functions will be added as well. Custom functions will now appear in the Custom Functions category of The Tree function for you to add to the formula. Page 12 Is one of the most frequently heard questions by users reporting Crystal 8.5 and earlier (especially users in larger organizations) Can I share parts of this report with other users or other reports? While some use creative copying and dough-off to the common word processor or text files may have provided a sliced answer to the question, the general answer was usually No. Perhaps the most anticipated new feature of crystal 9 reports, the repository, taken care of this need. The repository (further updated in version 10) allows multiple types of reporting objects to be stored in a central database to share among other reports and other users. Not only does this greatly reduce the need for repetitive reporting design steps, it also affords a simple, focused way to automatically update common parts of joint reports. The major tank change in crystal 10 reports relates to Crystal Enterprise 10 (often abbreviated as CE). While Crystal 9 reports independent repository database support, version 10 requires Crystal Enterprise 10 to be installed (the repository is currently stored in the CE 10 Crystal Management Server Database). While this change undoubtedly causes a grumbling among organizations that adopted Crystal Enterprise in the past, or which common repositories founded in independent SQL databases, it does not have a certain additional level of central management that you may find useful. The more detailed coverage point in Crystal Enterprise 10, its various versions and capabilities, and implementation scenarios, can be found in the second part of this book. Top 13 Simply Placed, Crystal Report Repository A common location where reporting objects can be saved. When you store a report object in the repository, it is available to add to other reports you create. If you connect to your common CE repository, which other users share in your organization, the objects you add to the repository will be made available to other users of crystal reports and added objects will be available to you. Version 10 repository now stores all your items in the CE 10 Crystal Management Server database. Any Crystal Reports 10 user who has a valid user ID and connection to your CE 10 system will be able to connect to the repository. Even if you have just a few reporting designers who share a shared repository, the benefits of designing a joint report and automatically updated reporting objects will soon be revealed. You use the repository in the Crystal Report with Tank Explorer, a separate frame (or un-dockable dialog box) inside the Report Designer, similar to The Explorer Field. When you initially attempt to display the repository explorer, you will be asked to first enter CE 10 (if you are not already log on). The repository explorer shows the repository classified into folders. Although the sample repository that ships with Crystal Enterprise 10 shows folders for specific types of objects (text objects, bitmaps, SQL commands), you don't need to organize folders this way you may put any type of object that the repository supports into each folder. You may add any of the objects in the repository to the appropriate areas of your report. The repository supports five types of crystal reporting objects: text objects are standard text objects containing static text (no embedded fields) Bitmaps (images) Bitmap images, such as logos, photos, and signature custom functions functions or subroutines that can be used within the reporting formula. Note that custom repository functions only appear in the formula workshop, not in the Repository Explorer. SQL Commands/Server Query is based on SQL commands that the report can be based on business comments (new in version 10) crystal enterprise-based meta-layer display database connection that the report can be based on (business comments are covered in more detail in Chapter 17.) text objects and bitmaps can simply be pulled from the repository explorer on the report, just as you drag and drop the field from Explorer Field. Custom repository functions should be added to the formula report from within the formula workshop (covered in Chapter 5), and sql repository commands and business comments should be added to the report from within the database expert (SQL commands are covered in more detail in Chapter 16, and business comments are covered in Chapter 17). Tip You can install sample objects into your Crystal Enterprise repository using the new Business View Manager Sample Sample Menu option. More information has been discussed about the use of the business viewing manager later this season, as well as in Chapter 17. The Explorer has been slightly enhanced in crystal report 10 by adding a toolbar, the ability to control a tidy screen of folders and repository objects, and the filter feature to limit the display of some tank objects. Repository Explorer toolbar includes four buttons to change display settings, advanced filtering, delete and add folders. The Delete and Add Folder buttons, described later in the chapter, perform relatively self-described functions. Advanced change settings and filtering buttons expose new features in repository version 10. Clicking the Change View Settings button, or right-clicking anywhere inside the Repository Explorer and selecting View Change Settings from the pop-up menu, the Display Settings dialog box will display. By checking any of the repository object types, you can remove them from appearing in the repository explorer. For example, you may prefer to see SQL commands (since you can add them to reports from Tank Explorer anyway). Just check the report commands to eliminate them from the repository explorer. You can also sort repository objects by name or type by clicking the desired radio button. If you click the Advanced Filter button, or right-click anywhere inside the Explorer repository and select Advanced Filtering from the pop-up menu, the repository explorer will expand to show two filter text boxes and one Apply button. Type in a combination of the full or partial name of the object in the Display items with this text in the Name box and the full author name in the items displayed by this author box. Then, just click the Apply button or just press ENTER. Tank Explorer replays showing only tank objects matching the criteria you typed. To remove criteria, just delete the text in the Filter boxes, and click Apply or press ENTER. If you no longer want to see filter text boxes, just click the Advanced Filtering Toolbar button again, or right-click and select the Advanced Filtering pop-up menu option. Note even though folders may show plus marks next to them after setting the filter, nothing will appear inside the folder if it does not meet the required filter. Page 14 When you install your initial Crystal Enterprise 10 system with its default database, you need to start populating it with your common objects. And while it's not needed, you'll probably gain more usability with the repository if you organize the repository into folders. You may create as many folders in the repository as you'd like. And you are not limited to creating a level of folders you may create subfolders inside other folders to create a hierarchy for repository objects. Once you've created your folder structure, you can place text objects, bitmap graphics, or SQL commands in folders (custom functions are placed in a separate part of the reserved repository only for themselves, they won't appear in the repository explorer). From within the Crystal Report, you create, Or change folders from repository explorer. Launch The Explorer Repository by clicking the Explorer Repository button on the standard toolbar, or view the Explorer repository from the select pull-down menus. If you have already entered Crystal Enterprise, you get to do so. If you haven't changed the repository yet, you'll see your server name at the top of the browser, followed by folders originally supplied by the OUT-of-the-box CE repository. You may create a high-level folder that appears directly under your CE server name, or a subfolder that exists in another folder. If you want to create a high-level folder, make sure you have selected the CE server name for the first time. Click the Folder button on the Repository Explorer toolbar, or right-click the CE server and select the new folder from the pop-up menu. A folder with the default name New Folder will be added to the repository, and you will be in edit mode immediately, where you can type your name for the folder. Type the name of the new folder and press ENTER or click elsewhere with YOUR MOUSE. If you want to create additional folders, you may create high-level folders by repeating the previous steps (just make sure you have selected the CE server before clicking the Folder Toolbar button). If you want to create an existing subfolder, select the higher level folder first. Then right-click the Folder button on the Repository Explorer toolbar and select the new folder from the pop-up menu. You will see a lower level folder in the higher level folder you right-clicked, again with the default New Folder name, which you can rename immediately. If you want to rename the folder you have already added, select the folder you want to rename, and then hold down the mouse button on the folder for a few seconds. It is also possible to select the folder and press F2, or right-click the folder you want and select Rename from the pop-up menu. In all cases, the folder name will go into edit mode and you can type a new name for the folder. Deleting folders is similarly direct. Select the folder you want to delete and simply press the DEL key. Or you may click the X button on the Repository Explorer toolbar or right-click the folder and select Delete from the pop-up menu. You will be asked to confirm this ruling if there are objects in the folder, they will be deleted along with the folder. Deleting caution folders, such as deleting individual objects in the repository, cannot be unsealed. Make sure the object or folder you want to delete is no longer needed before deleting it. You may add report objects to the repository at any time, even if you haven't created any folder structure yet (moving objects in or through folders will be discussed later in the season). The method of adding an object to the repository depends on the type of object you want to add. A bitmap text object or graphics can be simply added to the repository And drop it from the Design tab or preview the report into the Explorer repository. If you want to insert the object into a specific folder, drag and drop the text object or bitmap at the top of the folder you want to add to. You may also add a text object or bitmap by right-clicking on the object you want and selecting Add to Repository from the pop-up menu. Specify a name for the object to appear in the repository. Make sure you don't use the name used for another repository object of the same type in the same folder (you'll be warned if you do, and if you continue, you'll overwrite the repository object with the same name). But you can add a repository object of the same name to a different folder. Although not necessary, you can also add a author name and description for your login. If you select the pop-up menu method to add object, the repository browser will appear within the Add Item dialog box. If you want to put the object in a folder, select the folder you want to put the new object in. If you select the CE server name instead of the folder, the object will be placed directly under the server name in the Explorer repository. When you have specified all the information to add the object to the repository, click OK. If you wish to rename an object you have added to the previously added repository, you may simply select the object inside the Tank Explorer then hold down the mouse button for a few seconds. The object will go into edit mode, where you can type in a new name (if you try to use another object name of the same type and name in the same folder, you will get an error), and press ENTER or click away from the object name. The object will be renamed. If you initially place an object in a specific folder (or in any folder, but just below the server name itself), you may move or copy it to another location with a simple drag-and-drop operation. Select the object you wish to move and hold down your mouse button. Then, drag the object at the top of your preferred new location (a folder if you wish to move the object to that folder, or server name if you wish to move the object from a folder and place it just below the CE server). When you drop the object in its new location, it then appears where you dropped it. Hold down ctrl key while dragging the object to copy to your new location, also leaving the object in its original location. If there is already an object of the same name in the destination folder, you will be warned that it will be rewritten. Deleting a text object or bitmap from the repository is similar to deleting folders. Just select the object you want to remove and press DEL KEY. You may alternatively click the X button in the Explorer Repository toolbar or right-click object and select Delete from the pop-up menu. And as with folders, deleting text objects or bitmaps cannot be unseed. Make sure you really want to remove the object from the repository before Note that deleting an object from the repository will not remove it from the reports they originally used. It will still remain in those reports, but will not be available to add to any new reports. While you can add a SQL command to the report of the repository explorer (you have to do so in the database expert), you can rename it or remove it from there. Use the same steps mentioned earlier to remove or rename text objects and bits to remove or rename SQL commands. Note that if you remove the object from a repository that may be part of existing reports, users will receive an error message when they open the report and update the repository objects. After that, the object will appear on this report as last saved. Add custom functions from formula workshop to repository. By displaying the Formula Workshop, select the custom report function you want to add to the repository, and click the Add to Repository button. You may also right-click the custom function name and select Add to Repository from the pop-up menu. And you may also simply drag the custom function name from the custom report of category functions from the Formula Workshop to the Custom Repository functions category (drop at the top of each item in the category). Your attention will not be asked for the folder that will place the custom function in it. The folder structure discussed early only applies to text objects and bitmaps. Custom functions are placed in a separate area of the repository just for them. If you've assigned categories to your custom functions when you've created them, categories will be retained when you put the custom function in the repository. Removing a custom function from the repository is completely upsting. Simply expand the custom function category of formula workshop repository, select the custom function you want to remove, and press del key. You may choose to replace the right-click custom function name and remove from the pop-up menu. In both cases, you will be warned that deleting a custom function from the repository cannot be unsealed. Confirm that you want to remove the function and will be permanently removed from the repository. Note that deleting an object from the repository will not remove it from the reports they originally used. It will still remain in those reports, but will not be available to add to any new reports. Tip sample tank shipped with Crystal Enterprise 10 includes several samples of text objects, SQL commands, and bitmaps. These are just for demonstrations and will be low or useless when designing your own reports. There are, however, some useful custom functions supplied with the CE repository that you may find use in your reporting tasks. Add SQL commands to the repository in the database expert (creating SQL commands is discussed in more detail in Chapter 16). First, create the SQL command to appear in the list of selected database tables Then, right-click and select Add to Repository from the pop-up menu. If you haven't already entered CE, you'll be asked to do so. You will be presented with a dialog box similar to the dialogs described earlier, which will make you name the object and select a location in the repository for the object (SQL commands can be stored in the same folder structure as you set up in the repository explorer). You may optionally specify the author, as well as change the description for the SQL command to something other than the SQL statement that makes up the command (which is placed in the default description text box). When you click OK, the SQL command will be added to the repository. Then it will appear in the database expert repository section. Note you will also be able to see SQL commands in the Explorer repository alongside text objects and bitmaps they will only be displayed with different icons. However, you are able to drag and drop them to report as you can text objects or bitmaps. However you can rename and remove SQL commands from Explorer repository using the same techniques described for text objects and bits of gammas. You may rename a SQL command you added to the repository either in Tank Explorer or Crystal Enterprise Explorer that appears when you first enter the repository of database experts. The process is the same in all situations. Select the object, hold down the mouse button for a few seconds to put the object in editing mode, and change. You may also right-click and select Rename from the pop-up menu. Deleting a SQL command from the repository should be done in The Explorer Repository you can remove a SQL command from the repository in the database expert. When you enter the database expert package and report on the Design tab, display the Tank Explorer. Move the folder structure to find the desired SQL command. Right-click the SQL command name and select Delete from the pop-up menu, click the X button on the Repository Explorer toolbar, or press the DEL key. Note that deleting an object from the repository will not remove it from the reports they originally used. It will still remain in those reports, but will not be available to add to any new reports. Note the report of the new Crystal 10 view business is also stored and managed in the repository. Full details on how to maintain a repository based on business views are available in Chapter 17, creating and using business perspectives. Page 15 When you add objects to the repository as specified earlier in the season, you, along with other users sharing the same repository, want to add items to the report. Repository versions of text objects, bitmap graphics, custom functions, SQL commands, and business views are added in various locations throughout the Crystal Report. Add these items to your report of The Explorer Repository (which can be displayed by clicking the Explorer Repository button on Toolbar or by viewing the Tank Explorer pull-down menu item). Enter Crystal Enterprise if necessary. Then, simply expand each folder to reveal the text objects and bitmaps that are available. If you want to see the properties of a repository object, right-click the object and select Properties from the pop-up menu. A small dialog box will pop up showing different pieces of information about the object. To add the object to the report, simply drag it from the repository probe to the report. As when dragging and dropping fields from Field Explorer, the outline appears as you drag. Once you have reached the desired position for the object, release your mouse button. Text object or bitmap graphics will be placed on the report. Tip when unable to format or resize objects you don't worry about the report from the repository. These objects are connected to the repository and cannot be edited, resized or formatted until they are interrupted. This will be discussed later in the chapter. Use custom repository functions in the formula workshop. You may add a custom function to the report from the repository directly, which it is placed in the Tree Function Formula Editor for use such as built-in crystal reporting functions. Alternatively, you can use the expert formula to create a formula based on a custom function of the repository. To add a custom function to the report directly, so that it appears in the Formula Editor function tree, starts with the Formula Workshop view. You may do this by clicking the Formula Workshop button in the Expert Toolbar, or by selecting the Workshop Formula Report from pull-down menus. Expand the custom repository handle functions from the Formula Workshop until you find the custom function that you wish to use (you may need to enter Crystal Enterprise if you already have). Select the function you want to add, and click the Add Report Toolbar button in the Formula Workshop toolbar. You can also right-click the Name function in the Formula Workshop tree and select Add to Report from the pop-up menu. And finally, you may simply drag the custom function you wish to add to the report from the Custom Repository functions category functions to report custom tree category formula workshops. Since a custom repository function can potentially call other custom functions inside the repository, you may be asked to confirm in addition to reporting other custom functions. When you do this, all your custom functions from the added repository will now appear in the Custom Functions category of The Tree function for you to add to the formula. Tip more details about creating formulas with expert formulas and using custom functions in formulas can be found in chapters 5 and 6. Add repository-based SQL commands and business view to your report from database explorer, set-up location dialog box, or another data related to the dialog box that you normally use to add Tables to report. In these dialog boxes, you will see a repository folder that you can expand to see the folder structure that was set up in the repository. If prompted, enter Crystal Enterprise. Then expand the folders to reveal the SQL or Business Views commands located in the repository. An example of the database expert is shown here: Select the SQL or Business View command that you want to add to the repository. Then you may click the right arrow to add the SQL command of the selected repository to the report, right-click the command and select Add to Report from the pop-up menu, or just drag the command or business view from the left of the database expert to the right. Similar methods add SQL commands and business views to reports from the repository in other data-related dialog boxes. Page 16 As indicated earlier in the season, you may become confused once you add an object from the repository to your report. This may occur if you attempt to change the format of a text object you added from the repository, resize a bitmap graphic you added from the repository, or change the SQL command text or custom function that is added to the repository. In all of these cases, you will discover that objects are locked or set to a read-only status you will not be able to make these types of changes. This is behavior with design. In order to ensure tank controls the appearance and behavior of these objects, they remain attached to the tank when you add them to the report. As long as they are connected, the repository will control their appearance and size (in the case of text objects and bitmap graphics), or their text (in the case of SQL commands and custom functions). Probably the clearest clue that an object is connected is the inability to change it. However, in some dialog boxes, you'll also notice a small vertical bar icon next to the object name - which indicates that the object is connected to the repository. The main benefit of leaving objects attached to the repository

is to allow them to be updated automatically if any other changes to the repository. For example, if you have added a shared company logo and slogan to your report using a bitmap graphic attached to the repository and text object, you'll probably want all reports using those objects to automatically reflect changes in the repository should you change your logo or motto. In order to ensure consistency with this behavior, individual changes to objects when they are added to the report should not be allowed. Tip to ensure connected objects will be updated when you initially open your reports, you should make one of the two choices. By selecting file options from pull-down menus and checking updates of connected repository objects in the open option on the Report tab, you will make sure that all reports automatically update connected objects when opened. If you just want to update connected objects in the report by report Leave this global option off and instead check the Update Objects repository check box in the open file dialog box when opening the Crystal Report. To change this behavior and make manual changes to the added objects of the repository, you need to disconnect them from the repository. Start by right-clicking on the object. In the case of text objects or bitmap graphics, right-click the object directly on the Design or Preview tabs. For SQL commands or custom functions, right-click the command or function name in your respective dialog boxes. From the pop-up menu, select Disconnect with repository. Notice that you may now resize or modify text objects or bitmap graphics, as well as change the contents of SQL commands or custom functions. Also, you will notice that the attached vertical bar icon already listed is no longer visible. Be aware, though, that these objects will already behave as if you just added them to the report they have been automatically updated by the repository any more. Using the company logo and the example of the slogan mentioned earlier, if you cut off objects from the repository and save the report, they won't reflect any tank changes the next time you open the report. Continue with the company logo and examples of the slogan mentioned earlier in this chapter; Or, you may have a company-wide custom function that needs to be changed for a new fiscal year or a new rewards program. In this situation, you need to add the object from the repository to your report, make changes, and update the repository with the modified object. As mentioned earlier, you cannot edit or change the object attached to the repository. This may cause confusion if you are required to update the repository copy of an object. To get away with this confusion, consider the steps needed to update a repository copy of an object: add the object to a report from the repository. Disconnect the object from the repository. Make the necessary changes to the object. Save the object to the repository in the same folder and with the same name as the original object. Step 4 will show behavior that is slightly different from the first time you added an object to the tank. When you attempt to add an object to the repository in the same folder with the same name, you will be warned that an object already exists and given the opportunity to update the version in the repository with the version you are saving. If you choose to do so, the old repository object will be replaced with an updated version. When you do these steps, note that your updated object will only be added to the repository once again connecting you will be able to change formatting or contents. To make future changes to the object, you need to disconnect it from the repository once again. Note Business View Using Crystal Enterprise View Modified Business You can't update business views from inside crystal reports. The business show manager is discussed in detail in Chapter 17. Page 17 with a shared repository, as with company database connections and network resources, you may wish to limit the ability to update or remove repository objects to certain sets of key users, while letting the rest of those sharing the repository just add existing repository items to your reports. As Crystal Reports 10 repository is currently managed by Crystal Enterprise, you use the Crystal Enterprise user and group license to accomplish this. While the typical Crystal Enterprise user and the assignment of the group's right is carried out in the Crystal Management Console (more details are discussed in crystal enterprise architecture and management in the second part of the book), the repository rights are preserved in crystal enterprise's new business viewing manager. If you already have access to the business display manager in installing the existing Crystal Enterprise, you should specifically install the Business View Manager on a Windows PC from the CE product CD. And, needless to say, since the repository is already part of Crystal Enterprise, you should have a functional crystal enterprise system in place to use the business display manager. Notice other business viewing capabilities of the business viewing manager related to the rights of the repository in Chapter 17, created and covered using business perspectives. Start the Business Show Manager from the Windows Start menu. You will be asked to log in to Crystal Enterprise. Make sure you use a CE user ID (initially, an administrator ID or a member of the admin group will be required) that provides adequate rights to change the security of the repository. The Business View Manager will appear, with an anchored tank explorer window appearing inside it (similar in appearance to Tank Explorer in crystal reports). If you choose, you may unseat the repository browser and move it around as a free floating window. Notice that the structure of the repository folder is exposed in the Explorer repository: CE Crystal Management will appear its server name at the top of the folder hierarchy, with high-level folders appearing below them. If you click the plus sign next to a folder, it will expand to show any subfolder or repository objects within it. Notice the additional folder labels custom functions that you may not have seen before in other versions of Explorer Repository for the Formula Workshop. This is a dedicated folder inside the repository where all custom functions are held. As you can specify a folder name in the formula workshop when adding a custom function, all of them have been inserted into this single repository folder. Consider this repository folder and object hierarchy carefully when planning your repository security plan. If you simply want to secure a set of repository rights for the entire repository (maybe you want managers to complete Rights, but any other just to read the rights), you may set the rights of the repository at the crystal level of server management. However, if you want to offer more granular rights (maybe you want a sales group to sell full rights to the entire repository folder or just specific objects or subfolders in the sales folder), then you may specify the rights in the folder and object level. In addition, you want to consider the concept of inheritance rights, the ability to set higher-level rights in the repository (perhaps in crystal server management or high-level folders) and rights automatically inherited by folders, subfolders, and objects at lower levels. By creatively using different levels of rights and inheritance, you have very tight control (maybe more than you need) over tank rights. Start by taking the overall set of rights you want to repository as a whole; All folders, subfolders, and objects in the repository inherit your rights as set here. If, as discussed earlier, you're just willing to have a set of rights applied to the entire repository, you just need to set the rights at this level. Start by selecting a high level crystal management server name in Tank Explorer. Then, right-click and select Edit Rights from the pop-up menu. The Edit Rights dialog box will appear, as shown in Figure 7-1. Figure 7-1: Your Editing Rights dialog box is a list of Crystal Enterprise groups and users that have already been granted or stripped of rights at crystal management server level (the newly installed Crystal Management Server initially shows administrators and each group here). Note three different rights that can be set for each user or group: view the ability to add objects in the repository to report editing the ability to add new objects to the repository, or update and remove objects in the security setting repository ability to change the security settings of the repository for others more, if you cycle through the options available for each right by clicking the check box, you will find that right Can be granted, rejected or inherited. Granting the right expressly gives this right to the selected group or user. Denial of right explicitly distances the right from the selected group or user. And allowing the right to inheritance will grant or deny the right based on higher-level security settings (at crystal management server level, inherited rights only apply to the group of administrators, which automatically grant all rights to the repository). If you want to grant or reject the rights of additional groups or users specifically, you need to add groups or users to the list by clicking Add Groups or adding the Users button in the Edit Rights dialog box toolbar. These buttons add groups or add user dialog box showing all Crystal Enterprise views Or users (in fact to create additional CE groups or users, you need to use other CE administrative tools, such as the Crystal Management Console you can add this here). Use this dialog box to select an additional group or user to add to the Edit Rights dialog box. When additional groups or users that added let appear in the Edit Rights dialog box, you can grant or reject the rights of that group or user by clicking on the appropriate check boxes. The next time you set rights for crystal management server, new groups and users you have added will res pop up with certain rights you have granted or declined to be on their side. If you want to delete the group or user you have already set rights for, select the user or group and click the Remove Toolbar (X) button. Although it is not as relevant as crystal level server management, you may still want to see what the final or pure set of rights is for any group or user. Do so by clicking the Preview button in the Edit Rights dialog box toolbar. The list of users and groups will change and show the actual combination of rights arising from explicit settings and inheritance. If you need to change any rights, click the preview button again to return the list to its editable state. After determining the rights for all necessary groups and users, just click OK to close the Edit Rights dialog box. Your rights will be determined in the Crystal Enterprise system and the effect will be immediately stored for all repository users. The rights set at crystal management server level are automatically inherited by all folders and objects in the repository. Be cautious when denying rights at crystal management server level. If you deny a right to a group or user at this level, that user or group will not be right at any point in the repository, even if you explicitly grant it at a lower level. Because of the rules of reservoir inheritance, denial of rights has precedence over the granting of rights when rights are inherited. If you wish to use the Crystal Enterprise folder and the object at the security level of the repository, perform similar steps to those previously discussed to adjust the security of the repository at the crystal level of server management. However, before selecting the Edit Rights option from the pop-up menu, make sure you have selected the folder, subfolder or object you want in the repository explorer that you want the rights to apply. The same edit rights dialog box will appear. If it grants or rejects the rights of groups or users at a higher level, you will now see the group or user listed in the Edit Rights dialog box. You can either change inherited rights by clicking on the check boxes of existing groups, or add additional groups or users to the Edit Rights dialog box by clicking the Add Groups button or adding the Users button in the Edit Rights dialog box toolbar. The most important thing to consider when determining salaries at this level is inheritance. If there was a pre-right Or denied at a higher level (crystal management server or a higher level folder), which will be inherited right at this lower level. In particular, if a higher-level right has been denied, even granting it at this lower level does not give the group or user the right. If a right is not granted or denied at all, the granting at this level will enable the right of the group or user. If a right has never been granted specifically (whether at this level or at a higher level), then the user will not be entitled. Once you have specified the rights at a lower level, click the preview button to see the set of net rights granted to users and groups. The show takes into account the net rights of inherited rights from a higher-level folder and crystal management server, as well as the rights granted and denied specifically at this level. With the Preview button, you can see what each group and user will eventually be able to do inside this folder or to this object. If you need to change your rights after viewing the net rights, click the preview button again to return the list of groups and users to editable mode. Note preventing users from updating or deleting tank objects will still allow them to separate objects from the repository and change them when they are placed in the report. However, if they have any changes to such objects, they will still be able to update the repository with their updated versions. Page 18 was not managed inside the Enterprise Crystal when the tank was first introduced in Crystal Rips 9. The default repository installed with Crystal 9 report was a Microsoft Access database stored on the local C drive. If you wish to share the repository among more than one reporting designer, you needed to spot all users reporting Crystal 9 to a shared database (perhaps one stored in Microsoft SQL Server, Oracle, or other standard SQL database) where a shared repository database was kept. Crystal reports 10 tanks currently managed by Crystal Enterprise 10; If you have Crystal Enterprise 10 not installed in your organization, you will no longer be able to use the crystal tank. Instead of storing repository objects in a separate SQL database, they are now stored in the same database used by Crystal Enterprise Crystal Server Management. If you create a common repository database with Crystal 9 report, you need to migrate all or some of your version 9 tank objects to the new version 10 crystal management server repository for use with Crystal 10 report. Business Objects has supplied crystal tank migration wizard (a separate Windows application) with Crystal Enterprise 10 to accomplish this migration. If you have access to the complete installation of Crystal Enterprise, you will need to install crystal tank migration wizard to a Windows computer from the CE 10 CD program. Although you can install this transfer wizard on a computer with an existing Crystal 9 repository connection, you You don't need to just ensure that the computer has at least one ODBC data source that connects to the shared repository database. Note If you want to copy repository objects from one Crystal Enterprise 10 system to another, use the Crystal Import Wizard instead of the Migration Wizard. The Import Wizard will copy tank objects, along with other objects (such as reports) from one 10 crystal enterprise system to another system. Start crystal tank migration wizard from Crystal Enterprise 10 program group. Once you've read the introductory note, click Next to select the source repository. The drop-down source list either includes the name of your existing Version 9 repository (if you are running the Migration Wizard from crystal report 9 or Crystal Enterprise 9 computer) or a list of all ODBC data sources available on your computer (if your computer is running the wizard in to version 9 of the repository is not connected). Select the appropriate entry from the drop down list. If the computer you're using is connected to repository 9, you'll need to find only one entry in the list. Otherwise, select the ODBC data source name that points to your repository database. If you don't find the appropriate ODBC data source name, you'll need to run the ODBC manager from the Windows Control Panel and create a suitable ODBC data source for the repository database version 9. If the 9-member repository exists on a standard secure industry database such as Microsoft SQL Server, it specifies a user ID and password for the database. Click Next. Then you get to enter your Crystal Enterprise 10 Crystal Server Management. The supply of crystal management server names, along with user ID and password that provides editing rights to the repository (the setting of repository rights is discussed earlier in the chapter under the control of repository rights). Admin user ID or ID in the administrators group will normally provide adequate salaries. Click Next. The display of source repository objects will appear, showing all folders and objects in your Crystal 9 repository. By default, all objects will be checked, indicating that they will be moved to the new Crystal 10 repository. If there are any objects you don't want to migrate, check them out. Click Next. The Migration Wizard proceeded to copy all selected repository objects to crystal tank 10. The structure of the repository folder version 9 will be re-created in repository version 10 and objects of the same name will be copied. There should be duplicate objects already in tank version 10, they will not be rewritten with version 9 objects. The Transfer Wizard will show a check mark next to any successfully copied object. If any errors occur (maybe there are duplicate objects in crystal tank 10), different icons will appear next to the object. Click on the object name to see a more detailed description of the error. Page 19 A simple report is as fast as selecting tables, dragging and dropping fields onto the report, and clicking the Preview button. However, if you're doing only those few steps, you might have a much bigger report showing up than you bargain for! One of the most important missed steps is record selection. If you do not enter some record selection criteria, any records in the selected tables will appear in the report. In the case of a small computer-type database with, say, 1,000 records, this would be terribly time-intensive or resource-intensive. However, if you connect to a large SQL database with millions of potentially records, the consequences including record selection will probably be felt on your network, and they will definitely be felt on your desktop computer. Because crystal reports require storing data that makes up a report somewhere, you may run out of memory or temporary disk space in such situations. Regardless of these concerns, your report will be terribly slow, and probably won't be very helpful, if you have those many records in this report. Practically all reports will require record selection criteria. You may want to limit reports to only USA customers, only orders placed in 2001, or only invoices that have been due over the past 30 days. Record selection criteria can be used to limit your reporting to any of these sets of records. Either way, it's very wise to apply your record selection criteria early in the process of designing your report probably before you preview or print the report. Top 20 Crystal Reports provides expert selection to help you create useful record selection criteria. You can use the selection expert to select a simple, upstate, and as a starting point for a more complex record selection. The selection expert can run from the record selection section of each report wizard, or after selecting and linking the tables using the blank report option. Both way, you want to make sure you use it before previewing the report. To use Select Expert while using one of the report wizards, select at least one table in the Data section and link the Link section tables if necessary (the table link is covered in more detail in Chapter 16). Then progress through other sections until you reach the record selection section, which looks like this: If you are using the blank reporting option to create a report, you should first select and link the tables. When the design tab appears, you can either instantly run the selection expert or add fields to the report before running the selection expert. Again, you want to run the selection expert before you preview the report. Run the selection expert by clicking the Expert Selection in the Expert Toolbar. You can also select expert report from pull-down menus. If you already add fields to the report and want to use one of them from the Design or Preview tab to select the record, select the field on the report before starting the Select Expert option. You Also right - Click the field of your choice and select Select Expert from the pop-up menu. In each case, the selection expert will start with the worksheet already created for the selected field. Caution you may use this unwanted feature. If you start selecting Expert and seeing a tab for a field that you don't want to select on, just click the Delete button on the right of the Expert to remove the current tab. Then click the tab or <New>New button to select the correct field. If this is the first time you've run a selection expert, and you've not selected an existing report field, you'll see a Selection Field dialog box that lists all the reporting fields and databases. You will also see this dialog box if you are currently displaying the Selection Expert and clicking the new tab or button <New>to add additional criteria. Click the report field or database that you want your record selection to be based on. If you want to see sample data from the database for that field, click Browse. When you are satisfied with the field you want to use to select the record, select it and click Ok. The Select Expert will appear with a tab for that field. Once you have selected a field to select in, you will see an additional pull-down list with the default of each value. You will use this list to select the comparison operations you want to use to select your record. The pull-down list will reveal all the comparisons you can use to select records; Table 8-1 explains different operators that may appear in the pull-down list. Note that the pull-down list will include the operators discussed in the table, as well as operators like them except that they are not starting. As you might imagine, choosing the Not version of the operator includes all records where the situation is not correct, rather than where the bet is right. Table 8-1: Selecting expert comparison operators describe the operator each value is the same as having no selection criteria at all. Whether any value means that no matter what is in this area all records will be included in the report. The field should be exactly the same as what you specify. Is it one you can specify more than one item to compare by adding multiple comparison items to a list. If the field is exactly the same as any of them, the record will be included. Should be less than the field than in the case you compared to. If you are comparing numbers, the field should be numerically smaller. If you are comparing the date, the field should be the previous date. If you compare strings, the field should be lower in the alphabet. If you choose the option less than or equal, the field can be equal to or less than you compare to. Whether larger than the field should be more than the case you compared to. If you are comparing numbers, the field should be larger numerically. If you are comparing the date, the field should be a later date. If you are Strings, the field should be higher in the alphabet. If you choose the Greater Than Or Equal To option, the field can be equal to or greater than you compare to. Do between allows you to select two cases to create a comparison range. The field must be between two, or equal to one of them. Is Between uses the same type of comparison as is used with Is Less Than and Is Greater Than: numbers compare numerically, dates compare chronologically, and strings compareically. Starts by allowing you to specify the leading characters to compare with. If the first characters in the field are equal to the specified characters, the record will be returned. If you want to get multiple starts with comparisons, you can add multiple criteria to a list. This operator only appears when you use a string field. Do like you can to match the partial text using wildcard characters to search for records that include specific characters or groups of characters to follow. When you specify your comparisons, you can use a question mark to indicate that a character in this field in that position can contain anything. You can use a strick to show that the rest of the field from that point on can contain anything. If you want to make multiple such comparisons, you can add multiple criteria to a list. This operator will only appear when you use a string field. Is in the period allows you to compare a date field to a group of built-in date ranges, such as last week, last month, last quarter, this year, etc. These internal ranges are all based on your computer system clock when you run the report. This operator will only appear when you use a date field or date of time. It is true to include records where the field is equal to true. This operator only appears when you use a Boolean field. Does False contain records in which the field elysed with incorrectness. This operator only appears when you use a Boolean field. Formula allows you to insert any Boolean formula using crystal reporting formula language. Similar to the Show Formula button in the bottom right corner of Select Expert. Once you have selected a comparison operator in the pull-down list, the selection expert will change based on the selection you made. If you have chosen an operator that compares only to one item (such as equal, less than, or larger than), an additional stretch list will appear. If you have chosen an operator that can compare with multiple items (such as One Of, Like, or Starts With), a pull-down list will appear along with a multi-item box. You can add and remove items from multiple items box by clicking the Add and Remove buttons that appear next to the box. The new pull-down list allows you to select the item you want to compare the field with each of the two methods: you can type it directly or select it from the pull-down list. You simply can Literally you want to type in the pull-down list for comparison directly. If you click In the pull-down list, the expert will review the selection of the database and list a few item samples from that database field. You may choose one of the items in the pull-down list to compare. If the comparison action you selected allows multiple entries, you can add the item you typed to the multi-item box by clicking Add. If you select a reviewed database item from the pull-down list, it will be added to the multi-item box automatically. Both way, you can remove an item from multiple case boxes by selecting it and then clicking Delete. Note that you can choose not comparison operator versions either. This will actually reverse the selection criteria you have selected. If, for example, you have chosen a country field, chosen not equal to the operator, and specified the USA as an item to compare with, your report now includes records for each country except the USA. The expert of choice does not limit you to comparing just one field. Once you have added a database field, you can > > new tab or click new button. With this, select the context dialog box, from which you can select another field to compare. When you select this field, a new worksheet will be displayed in Select Expert, enabling you to create another comparison. You may create as many tabs and comparisons as you need. Crystal Tip report applies logically and to all tabs in the expert select all criteria should be correct to be selected for a record. If you prefer logical or applied to some or all of the tabs (so that if any of them are true, but not all of them, a record is returned), you need to manually edit the selection formula created by the selection expert. This issue will be discussed directly later this season in manipulating the record selection formula. When you preview a report on the screen for the first time, crystal reports actually read the database and perform record selections, and only then can it format and display the report. To increase future performance while you work with the report, crystal reports create a set of stored data. This stored data includes records recovered from the database, which are then stored on your hard drive, whether in memory or in temporary files. If you make simple formatting changes, move fields around, or make other minor changes that won't require database questioning, crystal reports will use stored data every time you preview the modified report, thereby improving performance. If you add new backgrounds to the report, Crystal Reports knows it needs to re-conquer the database, and it does so without evoking it. You may notice a bit of patience (or maybe a long wait, depending on your database) while it runs a new query. But when you change the criteria for record selection, crystal reports don't know if you need to Whether or not it has a database. You will be given the option to refresh or use stored data. Your choice depends on whether you are wider or <New > </New> Selection criteria. If you have narrowed your selection criteria so that the new selection criteria are fully satisfied with the available stored data, you can choose to use the stored data. Because the database does not have to be requeried, changes appear very quickly in the Preview tab. If, however, you will find the selection criteria wider so that the stored data contains all your records is not specified, you need to refresh the report so that the database can be requeried. Choosing to use the data stored in this situation will result in your report showing too few (if any) records, even if they are actually in the database. However, the new query will take time to do. If you make the wrong choice and end up with too few, or not, records, you can refresh the report manually by clicking the Refresh button on the standard toolbar, pressing the F5 key, or selecting report refresh report information from pull-down menus. When you save a report, you have the option to save the data stored in . RPT file . If it includes stored data, the report will immediately display the preview tab showing the stored data the next time you open. The RPT file will require no database requery. However, this also will. The RPT file is larger (sometimes significantly up), since it has to keep the data stored along with the report design. Tip Even if you open a report with saved data, the saved data will be discarded, and only the Design tab will appear if you discard the data stored in the open option in the File Options report tab. To choose whether to save data or not, check or mark the data storage file by reporting from pull-down menus, and then re-forgive the report after your selection. You can also make the selection with the appropriate check box in the file reporting options. If you want to set default behavior for this option for all new reports in the future, turn data storage on or off with the Report option in the File Options report tab. Many reporting requirements can be satisfied with the creative use of date fields in record selection. Crystal Report provides a good selection of built-in date ranges you can use to compare with, or you can use other operators to compare date fields. When you select a date field in the context dialog box, the selection expert makes available in the comparison operator period. If you choose this operator, another pull-down list containing crystal reports made appears in the date range. With these built-in ranges, you can create a report that returns, say, only orders in the previous month by comparison with LastFullMonth. What is particularly attractive about the use of date range functions is the self-maintenance of the report. When you use lastFullMonth range, for example, the report will always use the system clock on your computer to include orders from the previous month, no matter what the report runs. You don't need to change the date range manually every month. It could exist. However, when you need to manually insert a date range to select the record. If, for example, you want to see all 2001 orders, you need to specify those dates manually. There was no built-in X-year date range. In this case, you select the date field you want (e.g., order date) and use an operator between comparisons to show orders between January 1, 2001, and December 31, 2001. You can enter the start date of 1/1/2001 and the end date of 12/31/2001. Crystal Tip Report allows you to choose a fair amount of flexibility in date formatting with expert, generally allowing you to type in free form dates as you wish. However, you will still receive error messages if you type in a date that crystal reports are unsure about, such as dates that only include a double-digit year. If you are choosing in the date/time field with the operator between, supplying only one date, as opposed to the date and time, will work, but crystal reports will automatically assume time on or after midnight from the first date. However, only records that include exactly midnight times for a second date will be included will not be included every once, even a second after midnight for a second date. So, make sure you include a time value next to a date value if you want to select records based on some time other than midnight. Page 21 When you create record selection criteria with The Selection Expert, it actually creates formulas using crystal reporting formula language behind the scenes. For most simple record selection criteria, you have to worry about manipulating this formula directly. Also, using the selection expert directly and not manipulating the actual formula that it creates, you often maximize the function, especially when using the SQL database. However, there are times when the selection expert itself does not offer enough flexibility to choose the record you need to do. Consider the following scenario you have two fields included in your tabs in the selection expert: the area is equal to CO, and the order quantity is greater than 2500. Since the selection expert performs logically and between tabs, what would you do if you wanted to see all orders from Colorado, regardless of the order quantity, as well as orders from any other state exceeding $2,500? In this case, Select Expert does not offer enough flexibility to create this type of special record selection. Therefore, you need to use a record selection formula. The Select Expert create a record selection formula automatically as you add tabs and selection criteria. You can change its formula in one of three creation ways: click the Formula View button on your selection expert by selecting the record formula selection report from pull-down menus by displaying the Formula Workshop (discussed in more details in Chapter 5) and selecting the record selection in the Formula List selection category tip if you know you need additional features that the expert does not select. You can skip it completely and create your own record selection formula correctly in the Formula Editor or Formula Workshop. Select the record formula selection report from the pull-down menus or select the selection record in the Formula Bar, select formula formula formula formula formula formula this way. In the scenario mentioned earlier, you need to change the relationship between the two criteria from one And to one Or. This is a simple process that you can either apply right from the selection expert or using the formula editor. To use the selection expert, simply click Show formula. You can now change the formula created by the selection expert to show you all Colorado orders, regardless of value, and other orders exceeding $2,500. Note that Select Expert has placed the and operator between the two parts of the selection formula. Just put the locator in the formula and change and switch to Or, and then click OK. Tip If you click Formula View in the Selection Expert, and then decide that you want to use the full formula editor, just click the Formula Editor button in the Expanded Expert Selection dialog box. The formula is moved to the Formula Editor, where you can change or increase it. Since you will eventually be using crystal reporting formula language to select your record, many language features are available for record selection. In this situation, you may prefer to edit the selection formula in the editor formula so that you can see and use all the built-in functionality. The formula that the selection expert creates appears when you select the record of report selection formulas from pull-down menus. You can change this formula to your heart's content, provided that the final formula is finished Boolean formula it will finally only return true or false (refer to Chapter 5 to learn more in Boolean formulas). The formula for each record will be evaluated in the database. If the formula is evaluated correctly, the record will be included in the report; otherwise the record will be ignored. The sensitivity of the case with record selection is a question that you will probably ask yourself fairly quickly when using a record selection, is it a sensitive case? In other words, if you ask to see records where the country is USA, a record will be returned if the database field contains mixed case characters, such as the USA? Case sensitivity is generally ignored when using SQL databases and computer databases via ODBC, as well as some computer-style databases using a direct database driver. Although this is an ine senseless case of default behavior out of the box, be sure to check the inse senseless case database server option in the File Report Options dialog box to influence the current report, or check the same option on the Database tab of file options to set default for all new reports you will create in the future. Even if this option is checked, some databases and ODBC drivers may make a case of ine feelings with Crystal They don't. It's best to run a test with your database to make sure you recover all the records you want by selecting your record. Note if you change the formula you created the selection expert, or create your own formula, running the selection expert again is good. However, if the selection expert is able to fully interpret the formula you created, you will see slightly different behavior for one or more tabs. You may see a tab with a set field to formula Is and part of the selection formula showing in the third list box. You may also see a message indicating that the formula uses a compound phrase and prompts you to edit the formula directly. Page 22 When you use the Select Expert or create a record selection formula with formula editor, you will affect the way crystal reports initially select data from the database. Record selection occurs during the first report pass, before sorting or grouping data. Because of this, you can use record choices to limit your reporting, say, to groups where total sales of more than $100,000 record choices occur before this total is calculated (see Chapter 5 for discussion of the report passes). You may also want to use the existing report formula in the record selection. However, if you use the WhilePrintingRecords function or a summary function in the formula, it will be evaluated in the second pass of the report, and when you create a record selection formula, it will not be shown in the Field Tree box. Again record selection occurs during the first pass, and second pass formulas cannot be used. If you want to limit the report according to subtotals or group summaries, or somehow limit the report by using the second pass formulas, you need to use a selection group formula instead of the record selection formula. You may actually create a inadvertent group selection formula and you don't even know it. If you are using a subtotal field or summary you will create in your report on the selection expert (perhaps you choose total customer.Last year sales instead of the customer.Last year sell your database field), you will be using group selection instead of record choices. You may also create a selection group formula from the Selection Expert by clicking the Formula View and then clicking the Select Group Radio button. If you want to use a typed formula, you will have two selections: select the report selection formulas group from pull-down menus or select group in the Formula Workshop selection formulas category. You can now create a Boolean formula to limit records by using group summaries or second pass formulas. One word of caution: Group selection occurs after group tree, subtotals, and large sum is calculated. This can lead to obvious insetrus in your report. For example look at the report shown in Figure 8-1. Figure 8-1: The report using the Select Group you will notice that the Group Tree indicates many more areas than actually appear in the report. And he doesn't want a mathematical degree to see the big one. Don't quite add up. Don't forget that selecting a summary field or subtotal in Select Expert will create a group selection formula instead of the record selection formula. You may see this kind of strange behavior and don't fully understand why. Take an expert's look to see if your selection is based on a subtotal or summary field. The report applies a group selection formula to limit reporting to groups with sales totaling more than $250,000 last year. This group selection is applied after the group tree and large totals have been created. Although there is no way to change the group tree in this situation, you can correct the whole problem by using the whole running rather than the large total. Look at Chapter 5 for information on running in all fields. Page 23 In many cases, record selection is the most time-over-time part of the reporting process, especially with larger databases. If you are using a computer-style database located on a local hard drive or network, report Crystal performs your record selection, read each record in the database and keep only those matches. If you are using a server-based database (such as SQL Server or Oracle), Crystal Reports will attempt to create a WHERE clause in the query that is sent to the database server, which will cause the database server to conduct the query and send only the desired records to the Crystal report. In both situations, you'll probably see improved overall performance if you're using indexed fields to select your record (with computer-style databases, this is often very critical). Indexed fields are fields that are specifically determined when the database is designed. The field index stores all values in the field in a preclassified state that makes it much faster to select records by field. To determine if a field is indexed, you may want to consult with the database designer. You can also see which fields are indexed using the Links tab from the database expert. Click the Database Expert button on the Expert Toolbar, or select database expert database from pull-down menus. Then click the Hyperlinks tab (you must have added at least two tables to your report to see it). You'll notice a small tent character appearing next to each profile field (different colors you may see are generally insignificant to record the selection, as long as you select in a field with the symbol). Take note of the indexed fields and attempt to use them in the record selection. If the field you need to select is not indexed, and the record selection seems very lazy, you may wish to consult with the database designer about adding indicators for that field. Crystal Tip reports the version before 9 characters does not show the tent when using the SQL database. New crystal reporting versions now display index designations for popular SQL databases, as well as computer-style databases. However, depending on the database driver you are using, you may still determine the tent indicator in Finally, double-check the setting of usage indicators or servers for the Speed option. To check the current report option, see the File Report Options dialog box. If you wish to check the option for all new reports in the future, follow the option in the Database tab of file options. If this is off, Crystal Reports does not use field indicators at all. SQL databases (or PC-style databases accessed through ODBC) offer a different set of performance considerations than performance with PC-style databases. As a general reporting rule, you want to always do the database server record selection (via the paragraph where above), if at all possible. It can normally be done using only the selection expert to create selection criteria using the editor formula makes it completely too easy to introduce functions that crystal reports cannot move to the database server. Also, making changes to what the selection expert creates with the Formula Is operator or the Display Formula button may seriously degrade the database server record selection function. As with computer-style databases, make sure to use indicators or servers for speed options in clear file reporting options. The point is more in-depth discussion and examples of performance issues, including record selection, found in Chapter 16. Page 24 of the Crystal Report gives you considerable flexibility in customizing the appearance of objects that you put in your report, such as database fields, text objects, and formulas. With different formatting options for these objects, you can change many aspects of your appearance, such as font form, size, color, alignment, and more. The most basic type of formatting is known as absolute formatting, in which you simply select the object and make formatting changes with the Format toolbar or format editor. In both cases, the change will apply to all object occurrences in the report if you format a field in the Details section absolutely, that field will appear the same every time it prints. The quickest way to format one or more objects in the report is to select the object or objects you want to format and then select the options from the formatting toolbar. To select a single object, just click it with the mouse. To select multiple objects to format at once, CTRL-click or SHIFT-click on more than one object (you'll notice that all objects you've selected will have a shaded outline around them). Then click the buttons in the Formatting Toolbar to format the selected objects. Table 9-1 outlines each formatting toolbar button. Table 9-1: Formatting toolbar options button of Font Face function select different font faces (such as Arial, Roman Times, etc.) from the drop down list. Select the font size of the font size, in points, from the drop down list, or insert a value directly into the box. Increase font size increase font size (each click of this button increases the font size by a point). Reduce font size reduced font size Click this button to reduce the font size by a point). Bold format object uses bold emphasis. Italic Format the object using italic letters . Underline add a following line to the object. Left Align Align text to the left of the object's defined width. Center Align Align text to the center of the object's defined width. Right Align Align text to the right of the object's defined width. Full Justify Align on both the left and right sides of the object's defined width. It provides quite justified text, similar to that often found in newspaper columns. Change font color font color. If you click the button itself, it will adjust the font color to the one that is displayed in the small line in the button. If you click the down arrow, a dialog box will pop up that will give you a selection of colors. When you select a color that converts to the default color for the button, you will see the small line in the Change Color button. Outside the borders add boundary lines on the object side. If you click the face button, you will first be given to all four sides of the border object. If you click the face button again, the borders will be turned off. If you click the down arrow, a subset of buttons will appear that allows you to select combinations of borders left, right, up or down, all or none. Suppress the Toggle screen of the object on and off. This is equivalent to clicking the Suppression Check Box on the Format Editor Common tab. Format locks feature to change other formatting

properties on the object. If formatting is locked, all other formatting options, including width and height, will be disabled. This is equivalent to clicking the Read-Only check box in the Format Editor Shared tab. Lock size/position ability to change the width or height, or to move the object. This is equivalent to clicking the Lock Position and Size check box on the Format Editor shared tab. Display a currency icon (an icon selected as the default currency symbol in the Windows Control Panel) with the object. This button will only be activated if all objects you have selected are currency or numeric fields. Thousands of separator displays from thousands separator (symbol chosen as the default thousands separator in Windows Control Panel) inside the object. This button will only be activated if all objects you have selected are currency or numeric fields. Percentage of the sign registering the percentage mark on the right side of the object. This option actually adds a currency symbol, but changes the symbol to the percentage sign and position to the right of the object. This button will only be activated if all objects you have selected are currency or numeric fields. Increase the speed of increasing the number of places displayed. For example, if the object is displayed as $121.22 and this button is clicked, this number may be displayed as $121,223. This button will only be activated if all of your selected objects are currency or numeric A reduction of 10,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,00 This button will only be activated if all objects you have selected are currency or numeric fields, and if at least one tening location is already shown. Although the Crystal Report offers a large number of formatting choices in the formatting toolbar, there are still quite a few formatting options that you can do with toolbar buttons. For these formatting requirements, you need to use the format editor. Format Editor is not an editor per second, but a tabgraded dialog box that displays a different set of tabs, depending on the type of object data you format. To display the Format Editor, select objects as described earlier in this chapter, and then select one of the following options: select text format, field template format, or format format objects from pull-down menus (Select Text, Field, or Objects will be determined by the number and data type of selected objects before selecting the option). Right-click the selected object and select text format, field template or object template from the pop-up menu. Click the Format Toolbar button on the Expert Tools toolbar. The tab displayed in the Format Editor will vary depending on the type of object data you format. For example, if you had selected one or more date/time fields before displaying the Format Editor, the Date/Time tab will be displayed first. Number fields will lead to the Number tab display, and string fields or text objects will cause the paragraph formatting clause to be displayed first. If you select multiple objects of different data types, the Shared tab will be displayed by default, and you will be able to change only formatting options that apply to all selected objects: the data type of specific options will not be available until you cancel the format editor, select one or more objects with the same data type, and re-format them. In some format editor tabs (such as the Number tab or the Date/Time tab), you will be able to choose from the predefined formatting styles that appear in a list. This style offers more used formatting styles that you may choose with a mouse click. If, however, you'd like to use some combination of formatting that doesn't offer this style, a custom style button is displayed at the bottom of the Format Editor additional dialog boxes where you can format individual parts from the field, such as leading week days for a date field, or currency symbol for a number or currency field. Most format editor options are fairly self-explaning in terms of the type of data being formatted. For those options that are not self-explaning, crystal reporting online will help provide additional information. Also, additional discussion of different editor format options can be found throughout the book in sections and chapters related to different types Fields being formatted. Page 25 While the absolute formatting options in the format editor will solve many reporting needs, you will soon find that you may want to change the formatting object according to the data being displayed. This is called conditional formatting, which allows you to change the appearance of objects depending on their contents or contents of other fields, objects or formulas. Although conditional formatting facilities are limited only by your imagination and creativity, there are some immediate uses of conditional formatting that may come to mind showing sales figures in red if they are below a predefined level using a different font to highlight long-term customers adding a boundary around the invoice number if it's past because of the title showing of the report that is on the page The first differs from the rest of the graphical pages showing with the folder file icon whether an item file has been opened or closed is probably the easiest conditional formatting tool with crystal expert outstanding report, which allows you to arrange which appearance of a field will change when a specific condition is met. If a sales figure falls below a pre-started target for the section, you can have it stand out with the white font color in the red background. Or, you can change the border in the overdue day formula which is more than, say, 60 days. To use an outstanding expert with a field, select the field you want to change. Start the Highlight Expert by clicking the Highlight button in the Baccalaureous Toolbar, selecting the expert highlight format from pull-down menus, or right-clicking on the object and selecting the Highlight Expert from the pop-up menu. Figure 9-1 shows the outstanding expert. Figure 9-1: The leading expert expert's idea is to allow conditional formatting from a field without the complex knowledge of crystal formula language reports. With the drop down lists in the dialog box, you select a series of conditions (by clicking the new button several times), and selecting the specific formatting for each bet. Start by clicking the new button to add a new status. Then select the field you want to use in a comparative test from the first drop down list in the Value of the dialog box (you may select any field in the report for this test, not just the field you will highlight). Then select a comparison operator in the second drop down list. You will find many of the standard comparison operators you used in formulas or in the selection expert, such as less than, larger than, equal to, not equal, and the like. After doing this selection, enter a fixed value to compare in the third drop down list (you can also click the drop down arrow and select a value from the sample data in the list). Finally, select any combination of font and background styles, colors, and border styles that you want the field to display if the comparison is correct. For example, for the format of the sales shape to show as white In the red background if it falls below the pre-start sales figure of $1,000, select the field in the first drop down list, select Compare Less than, Type 1000, and then select the font color from white and red background. You will see a sample in the sample box on the lower right of the highlighted expert, as well as to the left of the currently created status in the item list box on the left. When you click OK, this field will show the white text on a red background for each sales figure of less than $1000. If you want to display more than one formatting option, you may want to set up multiple conditions. To expand in the previous example, suppose you want to show the sale of the bonus (over $5,000) in blue, in addition to the red background available to those under $1,000. Just click the new button under the item list box. You can enter a new bet and another set of formatting options. Both will apply to this field. You may have two conditions that conflict with each other. For example, you can have one condition that formats the contents of the field over $1,000 in red, and another that applies the blue formatting to contents over $5,000. Since both conditions meet more than 1000 situations, will everything exceed $1,000 (including anything over $5,000) in red? Depends on the priority that you assign the terms. If the status of over 1000 in the item list box is higher, everything will be over $1,000 in red. However, if more than 5000 statuses are set higher, then it has priority everything over $5,000 will be blue. Then, the second item on the list (more than 1,000 items) will be tested, putting anything over $1,000 in red. To change your preference, click the conditions you want to move, and then click the Priority up or down arrow. Page 26 Expert Highlighting is a simple and quick way to format fields, as you don't need to know the formula language to use it. However, trading is in flexibility. As your reports become more complex, sometimes the outstanding expert will not offer all the flexibility you need. For example, you may need to apply formatting other than just color and border. Or you may need to take a more complex test than that with comparisons that are in the expert. For these conditions, you need to use conditional formatting formulas. Conditional formatting formulas use formula editor to create one or more conditions to determine how the object appears. Before you learn how to set formatting conditionally, the important thing is to have a basic understanding of absolute formatting, which simply refers to applying normal formatting to objects with format editor. This type of formatting described in the previous chapter uses the Format toolbar or format editor to apply the same formatting to all field occurrences. If you right-click on an object and select Format Field from the pop-up menu, the format editor will appear. Then you can change Font, style, click the font tab, click, Or color. If you change the font color to red, all object occurrences on the report will be red. If you click the border tab and select the Drop Shadow check box, all object occurrences will have a drip shadow. This is the process of absolute formatting. The first rule to follow when it comes to conditional formatting remember that you should use the format editor. While you can perform absolute formatting with either formatting toolbar or format editor, you can set up conditional formatting only with formatting toolbar format editor not working. Approaching conditional formatting, it is important to distinguish between two types of format editor formatting features: multi-choice properties and silent properties. In the Font tab, fonts and colors are good examples of multi-choice properties. You can click a drop down list and select from any of the multi-fonts or colors. An example of a silent property is Drop Shadow on the Border tab, which only has one check box: it can only be turned on or off. Whether a property is a multi-choice formatting or off determining the type of formula you use it is conditionally adjusted. Multi-option properties are conditionally formatted with If-Then-Else or Select Case formulas, while the silent properties are conditionally formatted with Boolean formulas. Tip you need to get to know crystal language formula reports to use conditional formatting effectively. To refresh your memory, look for information about the If-Then-Else and Boolean formulas, as well as select the item in Chapter 5. To set the formatting conditionally, click the Conditional Formula button which is in the Format Editor next to the property you want to format. This format will display the formula editor inside the Formula Bar workshop (basically the same editor formula discussed in Chapter 5, but with a new title), shown in Figure 9-2. Note that you can set conditional formatting with either crystals or basic syntax by making your choice of syntax drop down list. If you are formatting a multi-option property, all options available for the property will appear at the top of the function tree box. If, for example, you are conditionally formatting the color property, you will see all the colors listed. If you are formatting a border, you will see different line styles available. Figure 9-2: The Format Formula Editor uses the If-Then-Else formula or selecting the Case formula to determine the object formatting. Your formula can be as simple or complex as you need. For example, you may have a formula to adjust the font color which is as simple as follows: If {Customer.Last Year's Sales} > 5000 Then Blue Else Black or formula to set a low boundary as complex as this: If {Orders.Order Amount} > 5000 and {Orders.Ship Via} = Fedex Then DoubleLine Else If {Orders.Order Amount} > 1000 and {@Ship Days} < 3 Then SingleLine Else NoLine Tip In Most Cases, you may type formatting values, As color or line type, go to formula directly (as previous examples). You may also double click the values shown in the Formula Editor function tree box. If you double click, the default cr will appear against the formatting value in the formula. With a few exceptions, either the name of the value in itself or the name of the pre-cr value is acceptable. You can use any simple or inked Formula If-Then-Else, or formula of the selection, as long as the results of each then, Else, or Case are one of the formatting properties found in the Tree Box function. When you have finished with formulas, you can use the Check button to check the correct formula syntax, or save the formula and close the Formula Editor format with the Save and Close button. The format editor will remain on the screen. Notice that the Conditional Formula button changes from blue to red, and the pencil character inside the button points at a different angle. This indicates that the conditional formula for this property has been set. To change the existing formula, click the Conditional Formula button again and change the formula that appears in the Format Formula Editor. To remove conditional formatting and return to absolute formatting (or no formatting at all), just highlight and delete all conditional formulas. Then click the Save and Close button. You will notice that the conditional formula button has returned to a blue color with the pencil pointed in its original direction. Note While most conditional formulas should use a built-in formatting function for your Then, Else or Case formula clauses, the Size property is slightly different from other properties. In this case, the result of your conditional formula should be a number showing the font size to be used. If you format a silent property, the general conditional formatting procedure is the same. But when you click the Conditional Formula button next to the property, you will not see any additional function in the Format Formula Editor function tree box, as you cannot use an If-Then-Else or Select Formula Case to format this property. Since the property can have only one of two modes, off or off, you need to format it with boolean formula which can return only one of two results: true or false. To add drop shadows to the customer name fields of customers who sold last year exceeded $100,000, start by right-clicking on the customer name field. Select the format field from the pop-up menu, select the Border tab, and click the Conditional Formula button next to the Drop Shadow property. When the Format Formula Editor appears, in the Boolean formula, type below: {Customer.Last Year's Sales} > 100000 Boolean formula will only be evaluated into one of two states: true or false. If the correct formula returns, the formatting property will light up and the field will have a drip shadow around it. If the incorrect formula returns, the property will be turned off The field will not have a drip shadow. You may be curious about how conditional formatting and absolute formatting interrelate. Consider the scenario below. You will select an absolute red color in the Format Editor font tab and click OK. Of course, every field event will be red. Then you return to the Format Editor and, without changing the absolute formatting, click the Conditional Formula button next to the Color Property, and add the following formula: If {customer.last year's sale} > 50000 then blue note is missing another paragraph. Remember that crystal report does not require another paragraph in the If-Then-Else formula. In a regular formula, if the If test fails and there is no other paragraph, the formula returns another blank string, zero or other default value based on the formula data type. But if there is another band and the absolute formatting is set to red, what color will the font have here? Contrary to what may seem logical, when the If test fails in this case, the font is displayed in black and white type, despite the absolute red formatting. This is with the design if conditional formatting is applied, the absolute formatting will be ignored. If the conditional formula fails (and there is no condition to get the failure, such as another paragraph), the default color or format of the Windows Control Panel will be used for that type of object. If you're not using Else's tabs, especially if you're formatting background colors, be careful. Black pen paint isn't necessarily problematic, but the background color of black will often cause your report to look like someone plastering an electric bar all over it! The exception to this rule, and the way to combine absolute and conditional formatting, is to use the DefaultAttribute function, which is located in the Formatting Function tree box category in the Format Formula Editor. If you use this function with then, Else or Case clause, the formula will use the setting of the absolute formatting property. Therefore, if {customer.last year sells} > 50000 then blue is another DefaultAttribute sales figures of more than $50,000 in blue and others to show in red (provided that the absolute color selected in the format editor is red). If you change the absolute color, then figures over $50,000 still show blue, but the rest in each color you specify as absolute. Tip If you formatting conditional on a field that also applies with the format highlight expert, the expert will give priority highlights. Conditional formatting will only be visible if it changes the formatting of a field. If you search through the format editor, you'll find that practically all formatting properties can be conditionally adjusted. One of the most flexible is the property of repression in the common tab. You may consider that perfectly regulating the suppressed property of limited usefulness. (Why don't you even bother putting the object on the report if you just want to suppress it?) There Some good reasons for suppressing the object; For example, the formula which specifies a variable to zero in a group header should be physically placed in the header to work properly, but you don't want zeros to be shown at the top of each group. There are many more situations where contingentally suppressing an object can be useful. Here are some examples, and boolean-related formulas you apply to suppressed property: put the word continue in the group header repeated in Chapter 3, the Repeat Group header was introduced. If you select this option in the Group Options dialog box, a group header section will be repeated at the top of a page if a group continues from the previous page. Indicates that this group continues from the previous page and adds readability to your report. Inserts a text object containing the continued word, or something similar, in the group header close to the group name field. You should now suppress it if it is not repeated in a group header. Contingentally suppress the text object with the following Boolean formula: Not the InRepeatedGroupHeader tip when you conditionally suppress the object, you are using the Boolean formula; When your formula returns correctly, the object will be suppressed. Therefore, you may have to think back when conditionally suppressed. Showing a single bonus message for certain records you may want a report showing that a specific record (e.g., a specific order or specific vendor) exceeds a predefined target value. Just create a text object that displays something like greetings! you exceeded the sales goal . Again, you should think about when you don't want the text object to appear, not when you do. Assuming a $10,000 sales target, conditionally suppress the text object with the following Boolean formula: {AccountRep.Sales} <= 10,000 showing a different title on page 2 and later you may want to have a larger report title, perhaps including the company logo and a large font, on page 1 of the report. However, every other page needs to contain a smaller title without the logo, and perhaps contain the continued word. If it has the large title in the report header, you will only see it at the beginning of the report. Inserting a smaller title in the page header, on the other hand, will lead to showing it on each page, including page 1. Create the text object which contains the smaller caption, and place it in the page header along with any column headings or other objects that you want to appear on each page, but conditionally suppress the text object containing the title with the following Boolean formula (PageNumber is a built-in function of the Print State Category Formula Editor function tree box that returns the current page number) : PageNumber = 1 tip You can also create string formulas that provide roughly the same functionality as these examples and place them in appropriate reporting sections. However, to minimize potential formula clutter, it may sometimes be preferable Just create text objects and conditionally suppress them. They.